Power User Taging System Based on Micro-service

Bo Zhang¹,*, Yin Liu², Songlin Zuo³, Huizhou Liu¹, Xiaowei Liang¹, Youxia Cao³, Yan Chen², Wen Zhang², Bao Yuan², Hong OuYang²

¹Anhui Electric Power Co. State Grid. Hefei, China
²Beijing Zhongdian Puhua Information Technology Co., Ltd. State Grid Information & Telecommunication Technology Group Co., Ltd. Beijing, China
³Anhui Marketing Service Center Anhui Electric Power Co. Hefei, China

*Corresponding author email: bozhang@sgcc.com.cn

Abstract. With the rapid development of the electronic channel of State Grid Corporation of China, the electric power user taging system, one of the back-end support systems of the electronic channel, is also expanding. In this paper, a micro-service-based power user taging system is proposed, and the micro-service function requirement analysis is introduced. The micro-service of the electric power user taging system designed in this paper run independently to meet the needs of daily operation, update and expansion of the application.

Keywords: micro-service, tag definition, tag scheduling, tag execution.

1. Introduction
With the rapid development of the Electronic Channel of the State Grid Corporation of China, a large number of registered power users have been accumulated, and the power user taging system, which is the support for the promotion of the channel, has been continuously expanded. The process coupling of the application of the monolithic architecture is relatively high, applications become more and more complex and larger over time, resulting in high development, extension, maintenance costs, low availability, and unfulfilled agile development and deployment changes. The micro-service architecture divides the single application into a group of small services, which are built around specific services and communicate with each other using lightweight communication mechanism[1-2]. Each service runs independently and can be updated, deployed and extended for each service to meet the daily running and updating requirements of the application. The micro-service architecture also makes the micro-service and the micro-service "loose coupling" in the structure, but in the function displays as a unified whole. The flexibility, scalability, scalability and high availability of micro-service are the inevitable direction of the future development.

Power user taging system based on micro-service uses decentralized distributed applications to support the expansion of the entire system without bottlenecks, as long as the development of business needs to expand. Finally, it can reduce the cost of research and development, reduce the cost of operation and maintenance, solve the performance problems and deal with the rapid changes.
2. **System micro-service function requirement analysis**

According to the overall analysis of the user tagging system, the system adopts the overall architecture of micro-service, which is composed of the business micro-service part and the basic micro-service part. Service micro-service consists of access data micro-service, Tag definition micro-service, Tag scheduling micro-service, Tag execution micro-service and Tag computing micro-service. The basic service consists of service registration and discovery micro-service and API gateway micro-service. The system functional architecture is shown in Fig 1 below. And the business micro-service functional architecture is shown in Fig 2.

![System functional architecture](image1)

**Figure 1.** System functional architecture

![Business micro-service functional architecture](image2)

**Figure 2.** Business micro-service functional architecture
(1) Access Data Micro-service: Data access as a stand-alone micro-service, its main function is to access the user's master data and behavior data, where master data represents the user's inherent attribute information, for example, name, gender, date of birth, region, etc, behavior data is the browsing history of the operation process, including information about a user browsing interface or a power purchase record related content.

(2) Tag Definition Micro-service: Mainly responsible for the definition of tags, storage Tag scheduling and execution of logical information. A complete Tag definition model should include: Tag Unique ID, Tag name, schedule Tag Execution Time, Tag execution logic, and Tag item and so on.

(3) Tag Scheduling Micro-service: Between Tag Definition micro-service and Tag processing micro-service, should play a link between the role of decoupling the definition and implementation of business.

(4) Tag Execution Micro-service: tag-execution micro-service mainly performs the tag-execution logic and stores the tag results, which should be decoupled from other modules in the whole tagging system. Listen for message queues after startup and receive Tag execution events. When the event is obtained, the execution logic of the tag is extracted and executed. Process the TAG execution results, group the tag results into a series of tag items, and store these tag items in the in-memory database system. Monitor the entire process. If successful, a success message is sent. If failed, retry. If failed after retry, a failure message is sent.

(5) Tag Computing Micro-service: Tag computing micro-service is a module that aggregates and outputs the results of a tag. Its purpose is to perform and / or non-operations on a number of tags that have been completed, get the set of users in the result (subpopulation) and the number of users in the result (coverage), respectively.

(6) Service Registration and Discovery Micro-service: Mainly used for each business micro-service to register itself at start-up, while getting the address and status of other micro-services in the cluster.

(7) API Gateway Micro-service: Defines micro-services for tags and provides a calling interface for Tag computing micro-services to interact with external services.

3. System Design and Implementation

The whole system is developed with Java language, the modules are developed with Spring Boot framework [3], which can be executed independently, and Spring Cloud framework [4] is introduced to make the modules micro-service, add Eureka service register and discover, and every module is automatically registered with Eureka Service, and every service sends a heartbeat to Eureka on a regular basis. Zuul service API gateway is introduced into the architecture, and external access to the cluster requires reverse proxy of the service using the API gateway. The middleware MQ introduced AMQP Messaging Protocol Rabbit Realizes Service Asynchronous communication [5] and encapsulates service communication in the form of events, which can be used to communicate between modules. In addition, the health status of the service is monitored by Tur something bine and Hystrix, and the fuses are properly handled and displayed when there is an exception. The system technical architecture diagram is shown in Fig 3.

3.1. Access Data Micro-service

From the analysis of system function, we can know that user data access micro-service is the data source of the whole system. The overall architecture of the access service is shown in figure. Access micro-service provides HTTP service for master data and behavior data, and every user data is sent to the URL address of the specified HTTP service. The system adopts Kafka to implement Asynchronous, high-throughput distributed publish-subscribe message system, and consumers can consume data asynchronously without affecting the production data of producers.
3.2. Tag Definition Micro-service

Tag definition micro-service provides the access to create a tag definition. Each tag consists of three parts: tag information, Scheduling Information and execution content. Each tag definition submits the three parts of JSON as the Body of the HTTP service to the user tag definition micro-service using the POST method. After the user Tag defined data is saved in the database successfully, the user Tag defined micro service sends a Tag submission event that encapsulates the Tag ID, Tag information, scheduling information, and execution logic, will be serialized in JSON form, sent out through the Rabbit MQ message middleware, retrieved and post-processed by the micro-service in the cluster that listened for the event.

3.3. Tag Scheduling Micro-service

Once the event is obtained, the Tag ID in the event is first extracted and checked in the scheduling system to see if the Tag ID is in the scheduled task table. If the Tag ID is already in the Quartz database table, indicating that the tag is already in the scheduled task table, delete the scheduled task for the tag, clear the thread for the Job, update the Quartz database table, and recreate the scheduled task. If the Tag ID does not exist, the scheduled task is created directly. Each schedule task is a tag task to be executed, and when a tag meets the execution criteria, the tag logic is not executed directly, but via the spring containers publish method, a Tag execution event is sent to the Rabbit MQ message middleware.
3.4. Tag Execution Micro-service

The Tag scheduling micro-service listens on the execution queue of the Rabbit MQ message middleware and triggers tag processing as soon as tag execution events are present in the queue. In the whole process of tag processing, it can be divided into three stages: tag logic stage, result classification stage and serialization storage stage.

1) Perform The Tag Logic Phase: get the tag execution event, get the tag execution logic from the event. Get Tag processor from the spring container. The instance object of the Java class, and calls the execute method to execute the proxy JDBC connection to execute the SQL statement and return a Set object. If an execution exception occurs, the exception is caught directly, the tag execution is aborted, and the failure event is executed using the publish send tag in the spring container.

2) Perform The Result Classification Phase: This phase classifies the Result Set results by tag items and gets a collection of user principal ids for each tag item.

3) Serialize The Storage Phase: After the Hash Map object of the Tag results, call the key Set method, traverse all the Tag item ids in the Hash Map, and get the Roaring Bitmap object of each Tag item, the serialization is stored in Redis.

3.5. Tag Computing Micro-service

The main function of user tag computing micro-service is to do complicated intersection and aggregation operation of tag items, which is mainly aimed at subdividing people in business. For example, micro-service definition defined by Tag has Gender Tag, Region Tag, Hobby Tag, and so on, and is stored as Key in redis with the naming rule of "Tag: Tag ID: Tag Item ID". The structure of case Tag is shown in Table 1.

| Tag ID | Tag Name | Tag Item ID | Tag Item Name | Key | Value (Main ID Set) |
|--------|----------|-------------|---------------|-----|--------------------|
| 1      | Gender   | 1           | Female        | Tag1:1 | 1,2,4,5          |
| 2      | Gender   | 2           | Male          | Tag1:2 | 2,3,4,6          |
| 2      | Region   | 3           | Beijing       | Tag2:3 | 1,3,5,7,9       |
| 4      | Region   | 4           | Hangzhou      | Tag2:4 | 2,4,6,8,10      |
| 3      | Hobby    | 5           | Painting      | Tag3:5 | 1,2,3,4,5       |
| 6      | Hobby    | 6           | Singing       | Tag3:6 | 6,7,8,9,10      |

The first part is the parsing of tag expressions. The infix expression is processed in reverse Polish to get the postfix notation. The second part decomposes the postfix notation into a stack one by one, and performs a set operation on the set of two Tag items. Finally, the operation result of Tag expression is obtained, and the two parts can be dealt with effectively in the process of Tag calculation.

3.6. Service Registration and Discovery Micro-service

Micro-service registration and discovery uses Netflix's open source Eureka Service Registration and discovery, which requires the introduction of Eureka clients into every business micro-service. When the micro-service starts, the client provides service registration and other service status and address information.

3.7. API Gateway Micro-service

The API gateway micro-service also uses Netflix's open source Zuul middleware, which provides external accessible HTTP proxies, business micro-services that expose HTTP services via Zuul, and load balancing via Zuul gateway services, reduce service pressure on business services.
4. Conclusion

With the rapid development of Internet technology, traditional single architecture has been difficult to meet the development of Internet technology, micro-service Architecture emerged as the Times require. Compared with single architecture, micro-service architecture can simplify complex problems, has strong ability of lateral expansion, and the services are independent and uncoupled. Power user tagging system based on micro-service uses decentralized distributed applications to support the expansion of the entire system without bottlenecks, as long as the development of business needs to expand. Finally, it can reduce the cost of research and development, reduce the cost of operation and maintenance, solve the performance problems and deal with the rapid changes.

References

[1] ES Thönes. Micro-services [M], IEEE Software. IEEE. Volume: 32, Issue: 1, Jan.-Feb. 2015.
[2] Jiang Biao. DOCKER. micro-service architecture. Publishing House of Electronics Industry, 2018.
[3] Wang Yonghe, Zhang Jinsong and others. Research and application of Spring Boot. Information Communications, 2016, 10: 91 - 94.
[4] Jang Fangxu. Design and implementation of business system micro-service based on Spring Cloud. Electronic Technology and software engineering. 201808: 60 - 61.
[5] Yu chaowei, Zhan shubo. Implementation of asynchronous full duplex message bus J based on Rabbitmq, 2016, 02: 139 - 146.