Application of microservice in electric power unified modeling platform

Qiang Yang*, Shanyi Xie, Shu Huang and Ziying Zhang

Electric Power Research Institute of Guangdong Power Grid Co., Ltd., Guangzhou 510080, China

*Corresponding author’s: yangqiang0101@163.com

Abstract. Electric power unified modeling platform based on the traditional monolithic architecture has some disadvantages of monolithic architecture application, such as limited scalability and inflexible upgrade and deployment, etc. This paper proposes how to apply microservice to solve these problems, and make the platform more flexible and efficient in the process of data application, function extension, upgrade and deployment. The integration method of microservice and existing service architecture, and the implementation strategies of identification, deployment, registration, discovery and security access control of microservice are introduced in detail. The practical application of microservice in task management, data analysis, and data access has verified the effectiveness of the scheme. The application of microservice enriches the service category of the platform, simplifies the data access mode, and enhances the convenience of interaction with the platform.

1. Introduction

Electric power unified modeling platform is a platform that integrates data of multiple power automation and information systems. According to the data integration scope of the platform, the system data of dispatching automation, safety production, marketing, online monitoring, customer, environment, geographic information, etc. are standardized and integrated [1]. Data access support for the application is provided by standardized services including component interface specification, OPC unified architecture and enterprise Web services [2–4]. The established standardized services can support data access and system management as a whole, but they have relatively high technical complexity and high degree of coupling, which increases the difficulty of data application to a certain extent, and makes it difficult to extend personalized data access service.

Microservice is a software design idea and architecture, which aims to decompose the big functions into small service modules. Each service module can be deployed and operated independently to minimize the coupling of the system and reduce the interdependence of modules, so that it can be upgraded and deployed more flexibly [5]. With the characteristics of small module, distributed deployment and easy expansion, microservice is suitable for decoupling monolithic application, and also can be used as the basis of application development. Literature [6] introduces how to build a unified application development platform based on microservice.

However, for the electric power unified modeling platform and various data platforms in operation, the ideal way to apply microservice is to make it a supplement to standardized services. This paper studies the incremental introduction of microservice to the unified modeling platform. Microservice is coordinated with the existing service management of the unified modeling platform, and meets the...
technical requirements of the platform from deployment, discovery to access control, and is concretely applied to the data collection task management, personalized data service provision, etc.

2. Platform service architecture integrating microservice

The microservice of the electric power unified modeling platform is an extension of the existing standardized services, and the platform service architecture including microservice is shown in figure 1.

Through the network service management module of the platform, the internal application of the platform establishes the connection and interacts with the service after obtaining the access point information of the target standardized service.

![Diagram of Platform Service Architecture](image-url)

**Figure 1.** Microservice and electric power unified modeling platform.

The platform's service architecture includes the existing standardized services and the extended microservice. All modules and services are managed by the platform's network service management module (XAgent). Standardized services include component interface specification (CIS service), OPC UA service and enterprise standard defined Web services, providing object data access services based on classification, view and template, involving data of model, real-time, history, event and so on. Microservice is a supplement to standardized services that provides service support beyond standardized services, such as newly added system maintenance (data quality analysis, data tracking, etc.) and personalized data access, etc.

All microservices form a hierarchical microservice cluster as a whole. The microservice in the upper-layer can call the microservice in the lower-layer, but the microservice in the lower-layer can't call the microservice in the upper-layer, and the microservices in the same layer can't be called by each other. The dependencies between microservices form a tree structure, which is actually a one-way dependency. This can help to rationally partition the functionality of the microservice, avoid excessive dependencies and prevent the microservice system from losing its flexibility of deployment.

The internal application of the platform can either use standardized services after obtaining target services through XAgent, or directly access any microservices in the microservice cluster. The external application can't directly access the microservice cluster and must through the microservice proxy. The microservice proxy can provide a unified and stable access point for the external application, and also facilitate the security control and information statistics of external access by the system.

2.1. Network service management

The network service management module (XAgent) runs on each computing node to form a unified network management environment serving other modules of the platform. According to the module and service configuration of the whole network, XAgent on the main management node dynamically
determined by the election algorithm determines the running state of the service (master running, slave running, balanced running, etc.). Each module which has implemented the standard service provides service in master/slave or parallel mode under the coordination of XAgent.

Microservice is also managed by XAgent. Each microservice registers in XAgent as a separate module and publishes its own service address. By registering with XAgent and providing discovery information to the application, the unified modeling platform is equivalent to using XAgent to implement the discovery of the microservice.

2.2. URL path organization
The URL path organization of the microservice interface is implemented by referring to and moderately modifying the URL organization and request way of the RESTful. These mainly include extending HTTP verbs and redefining the way by which the URL organized.

Extending the HTTP verbs to specify the operation of resources, such as the actions that are not purely for the acquisition, submission or update of a resource, but for refreshing resources, performing tasks, login, clearing cache, etc.

The address of the microservice interface is organized as: "/category of the function[refined classification of the function][parameter qualified by upper-layer]/Noun/Verb[[name of parameter for Noun]/value of parameter for Noun]". Contents in "[]" are optional. Table 1 shows URL examples of two interfaces of GET and POST request methods.

| No. | Request method | URL | Category of the function | Refined classification of the function | Parameter qualified by upper-layer | Noun | Verb | Name of parameter for Noun | Value of parameter for Noun |
|-----|----------------|-----|--------------------------|----------------------------------------|-----------------------------------|------|------|--------------------------|--------------------------|
| 1   | GET            | DataProvider/Substation/objects/get/ofRegion/[regionName]/ofVoltage/[voltage] | Substation | objects | get | ofRegion, ofVoltage | [regionName], [voltage] |
| 2   | POST           | maintain/data/object/commit/ofType/[cimClassName]/ofURI/[uri] | maintain | data | object | commit | ofType, ofURI | [cimClassName], [uri] |

The interface in example 1 is used to query substation objects with a specified voltage grade in a certain region. The interface in example 2 is used to submit the specific type of object data to the backend model server, which doesn't include the property value information of the object in the URL.

When an interface of the microservice provides services to the external application through the proxy, the exposed interface will add module name. The interface in example 2 is in the data maintenance microservice module which named "MS-DataMT", so the exposed interface is "/MS-DataMT/maintain/data/object/commit/ofType/[cimClassName]/ofURI/[uri]".

3. Operation management of the microservice

3.1. Service multi-instance, load-balancing and non-stop service update
The same microservice can run instances on multiple different machines at the same time, and each instance can register module to XAgent. Multiple modules with the same name, in which only one is the master, and others are slaves. When the master module quits or switches to slave, XAgent automatically elects one of the slave modules as the master. In general, a microservice only needs to run one instance. When load balancing is not enabled, the master microservice can be accessed directly if there is no special designated slave microservice as the access object.
When a single microservice instance can't carry the access load, it is necessary to run instances on multiple nodes and turn on the load-balancing mode. In load-balancing mode, all microservices with same name are considered as the goal of load-balancing management.

Through load-balancing, access requests are allocated to each instance according to certain rules. The proportion of the load can be artificially set according to the performance of the node and the available computing resources. Since the microservice obtains service access points through XAgent rather than fixed IP, the load-balancing is controlled when getting of the access point. Assuming that there are three instances, the apportioned proportion are P1, P2 and P3, they are the numbers in the interval (0, 1) and are added up to 1. Use the random function to generate values called "randValue" between [0, 1), and the corresponding relationship between randValue and accessible instances is shown in table 2. The information about whether the load-balancing is enabled, and about the allocation proportion is stored in Zookeeper, which can be read in real time by each node of the system.

| Range of randValue | Accessible instance |
|--------------------|---------------------|
| [0, P1)             | Instance 1          |
| [P1, P1+P2)         | Instance 2          |
| [P1+P2, 1)          | Instance 3          |

Both the master-slave mode and load-balancing mode support uninterrupted service upgrades. When upgrading the microservice running in the master-slave mode, slave is updated first. When the update has completed, the slave will be switched to be the master, then update the original master to ensure the service is continuous. Services running in load-balancing mode can be upgraded one by one when the load is relatively low.

3.2. Request source information record and statistical analysis
Requests received by microservice may come from the platform's internal application modules, or from external access forwarded via the microservice proxy or web server. In order to analyze and monitor abnormal service access, the source of the request must be clear. Therefore, when making a request to the microservice of the unified modeling platform, three headers need to be added at the request header:

1) XZ-Module, the name of the module that makes the request. If it is forwarded by multiple levels, module name is appended in turn, separated by ";";

2) XZ-RootIP, the original IP that makes the request. Such as the IP of an external application, the host IP of browser that makes the request;

3) XZ-UserNme, the user name of authentication user who makes the request. Microservice records the request information in the big data database in the form of log. By analyzing these logs periodically, it is possible to count the number of visits per hour and day of each microservice and the number of visits per IP or user, and to analyze the abnormal visitor. For abnormal visitors, the administrator can remind them to make an explanation, or modify programs, or block them.

3.3. Secure access control
Microservice only responds directly to system internal module requests from hosts collaboratively managed by XAgent. After verifying the information of access source, microservice opens services only for internal requests. External application access must be done through the microservice proxy, which is an internal module of the system and runs on the system node. When accepting requests from external applications, the microservice proxy is responsible for the secure access control of external applications. Three security authentication mechanisms are used in the process of secure access control:

1) IP-based. Identify the identity of the external application by IP;
(2) Session within the validity period based on username/password login. Multiple requests can be made during the validity period after session login. External applications can extend the validity period of session by submitting authentication again before the session expires;

(3) Dynamic password based on valid time limit and IP limit of username/password authentication. When submitting a request to the microservice proxy, a dynamic password is included in the URL. This dynamic password can only be used by the module on the host that issued the password request, and must be reapplied after the validity period.

After authentication is passed, the microservice proxy will perform further interface licensing and data license verification. That is to verify whether the user has the right to access the requested interface and the target data. The granularity of authority control from coarse to refined is as follows:

1. Module-level. Define a microservice as a permission;
2. URL path matching. Use "**" or regular expression matching. "**" matches any string that doesn't contain "/", "***" matches any string with or without "/";
3. Interface-level. Define each method interface exposed by the microservice as a permission;
4. Parameter-level. The definition of parameter-level permissions depends on the expression language after the extension of the IK expression [7]. Take the acquisition of the interface of all substation objects with the specified voltage level under a power supply bureau as an example. The URL of the interface is: "/DataProvider/Substation/objects/get/ofRegion/{regionName}/ofVoltage/{voltage}" ("{}" indicates the needed parameters), and the limit condition for restricting "regionName" to "XX Power Supply Bureau" and for restricting the voltage level to 110kV or 35kV is: "$\{regionName\}=="XX Power Supply Bureau" && $arrayContains ("110kV"#, "35kV", $\{voltage\})".

Such as the limit of the object URI or the region to which the target object belongs can be used to define the limit conditions.

4. The application of the microservice

4.1. Task management-control
Various tasks such as data collection, data analysis, and data calculation are important components of the unified modeling platform. The configuration, status, and other data of these tasks (including the data source used by the task, the running log, the encoding basic information constructed when the model is imported, the object evolution information, the aggregation-analysis results, etc.) need to provide the interface for other applications to query. Before the microservice interface is provided, the data needs to be acquired based on the task-related client code, which makes logical implementation complex and maintenance difficult.

Each request of the microservice is a stateless remote call with clear semantics. The service definition is oriented to the application's final requirements rather than the original data. The interface of the microservice targeted at different levels of requirements, including the most basic single data object access, as well as obtaining complex analysis and processing results. Applications use the HTTP protocol directly for requests without relying on client-side code, which is useful for extension. The application-side only needs to put forward requirements, call interfaces, and apply data, so that makes the previously decentralized task management-control implementation can be focused on the microservice. The task management-control with the microservice architecture can completely decouple the application-side and the control logic, so that the upgrade and extension can be performed on demand.

4.2. Data analysis
As a platform to provide external data services, the unified modeling platform takes data as its core, and the data quality is very important. The platform data is collected from various business systems or collected from the terminal [8]. There are many quality problems, which need to be discovered through data analysis. Data analysis can be performed either periodically or manually. The periodical analysis is performed automatically by periodical tasks according to configuration, and the manual
analysis is performed by execution-side designed as a microservice after setting various parameters manually on the user interface. The data analysis microservice provides functions such as analysis configuration, analysis execution, acquisition of summary information of periodic task analysis results, task monitoring, configuration, and scheduling plan modification. The design of microservice interface corresponding to the manual-controlled user interface greatly facilitates the implementation of the user interface and the addition of data analysis functions.

4.3. Simplified data access

Microservice provides simplified data access for applications outside standardized services. The simplest case of microservice based on the HTTP protocol is to return the human-readable and machine-readable result data by filling in the correct parameters into the URL, which effectively reduces the learning cost and error probability, and greatly lowered the application threshold of platform data.

Unified modeling platform provides support for generic and specific data access using object access services provided by the microservice. For example: the interface "/DataProvider/Substation/objects/get/ofRegion/XXX power supply bureau/ofVoltage/110kV" is used to obtain the 110kV substation objects under XXX power supply bureau; the interface "/DataProvider/supplyPath/get/ofConductingEquipment/K003041245002" is used to obtain the power supply path (power supply path query function) of the conductive device coded K003041245002.

5. Conclusion

This paper studies the application of microservice to the unified modeling platform. Microservice is managed by the network management service of the platform, and microservice discovery is realized by the existing service discovery mechanism of the platform. Microservice is integrated with standardized services of the platform and has gained substantial application in task management-control, data analysis, and simplified data access. With the addition of microservice to the unified modeling platform, the coupling degree of modules is lower, the scalability of the platform is stronger, the secure access control is more detailed and accurate. The easy-to-use interface also reduces the development difficulty, and improves the capability of the platform data management and service, and the convenience of the platform application development.

Acknowledgments

This work was financially supported by the Science-Technology Project of Guangdong Power Grid Co., Ltd.(GDKJXM20173017).

References

[1] Xie, S.Y., Yang, Q., Liang, C.H., et al. (2016) Power transmission and transformation equipment management-control data platform based on global unified modeling. Guangdong Electric Power, 29(1): 51-57.
[2] Ma, Q., Jing, M., Yan, F. (2011) Standardization Design and Implementation of Integrated Data Platform for Power Dispatching. Electric Power Automation Equipment, 31(11): 125-129.
[3] Luo, N. (2014) Design of CIS interface for transmission and distribution network analysis platform. Xi'an: Xi'an Shiyou University.
[4] Mao, P., Li, X., Qin, H., et al. (2010) Design of Data Access Service for Public Information Platform. Electric Power Automation Equipment, 30(10): 121-125.
[5] Wang, J.J., Zhang, B., Gu, Y.S., et al. (2017) Microservice Architecture Evaluation of Web Applications in Cloud Environment. Application of Computer System, 26(5): 9-15.
[6] Li, C.Y., Liu, D., Cui, W., et al. (2017) Unified application development platform based on microservice architecture. Application of Computer System, 26(04): 45-50.
[7] TR_SMC. (2013) Introduction to IKexpression. https://blog.csdn.net/o9109003234/article/details/13293779.
[8] Xie, S.Y., Xu, H., Cai, H., Yu, Y.L. (2018) Research on Multi-source Data Integration in Data Platform of Power Distribution and Utilization. In: International Conference on Manufacturing Technology, Materials and Chemical Engineering. Zhuhai, China.