Research on Micro Service Architecture of Power Information System Based on Docker Container

Lin Qian\textsuperscript{1,a}, Haiyang Chen\textsuperscript{1,b}, Jun Yu\textsuperscript{1,c}, Guangxin Zhu\textsuperscript{1,d}, Jiyang Zhu\textsuperscript{2,e}, Chunlei Ren\textsuperscript{2,f}, Zhu Mei\textsuperscript{1,g}, Hengmao Pang\textsuperscript{1,h}, Mingjie Xu\textsuperscript{1,i}, Lin Wang\textsuperscript{1,j}

\textsuperscript{a}qianlin@sgepri.sgcc.com.cn, \textsuperscript{b}chenhaiyang@sgepri.sgcc.com.cn, \\
\textsuperscript{c}yujun@sgepri.sgcc.com.cn, \textsuperscript{d}zhuguangxin@sgepri.sgcc.com.cn, \\
\textsuperscript{e}zhujiyang@md.sgcc.com.cn, \textsuperscript{f}renchunlei@md.sgcc.com.cn, \\
\textsuperscript{g}meizhu2016@aliyun.com, \textsuperscript{h}panghengmao@sgepri.sgcc.com.cn, \\
\textsuperscript{i}xumingjie@sgepri.sgcc.com.cn, \textsuperscript{j}wanglin18@sgepri.sgcc.com.cn

Abstract. With the development of energy internet, the complexity of power information system is increasing day by day, and the ecosphere of cooperation and sharing among different industries is gradually formed. Based on Docker container technology and Kubernetes container layout technology, this paper studies the design principle and top-level architecture of power information system micro-service platform, and proposes the implementation scheme of service discovery, message communication and load balancing in power information system micro-service. Finally, the performance of the system is verified by simulation experiments.

1. Introduction

With the continuous development of information and communication technology, the application environment of power information system has undergone tremendous changes. In many business scenarios, the business application of single architecture is becoming larger and larger, the code logic is becoming more and more complex, and the coupling degree between codes and modules is getting higher and higher. Any modification of local functions will lead to the recompilation and deployment of single application, and the iteration cycle is long and unfavorable. Frequent deployment results in poor flexibility, weak sustained delivery capability and difficult maintenance of the entire business application, which is unable to respond to the sustained changes of the business in a timely manner. However, the traditional data center based on server cluster has many disadvantages, such as low resource utilization, complex management and maintenance, difficult migration of application services, and long start-up and shutdown time, which seriously reduces the user experience.

Microservice is a method of developing a single application as a set of small services. Each application runs in its own process and communicates with a lightweight mechanism. Micro services are built around business functions, support different programming languages, use different data storage technologies, and can be deployed independently through fully automated deployment mechanism.

Under the micro-service architecture model, traditional single power information system applications can be divided into several fine-grained micro-services and micro-applications that can be independently developed, compiled, deployed and operated according to business and technical needs. Each micro-service and micro-application code is small, independent, easy to maintain, compile and
deploy, and the development iteration cycle is short. Due to the small volume of deployment packages, fast service start-up and resource recovery, and easy to achieve flexible scaling, it can well meet the application scenarios with high concurrency and large fluctuation of access. In addition, the separately deployed micro-service barriers are well isolated, and the problems of single micro-service and micro-application will not lead to the unavailability of the whole business application. With the above advantages, power information system uses micro-service to develop business applications. Business applications can achieve gray publishing, frequent deployment, strong sustained delivery ability, and flexible and fast to adapt to the continuous development of business changes.

Docker is a lightweight virtualization solution that improves the running speed of the container and reduces the consumption of memory and other resources by sharing the kernel between the virtual container and the host. A Docker container is equivalent to a virtual machine, which can either have the functions of a specific operating system or deploy a specific application. Compared with common virtual technologies such as KVM, Docker's most obvious advantages are faster creation speed, faster start-stop speed, less resource occupation, and more flexible and convenient business deployment. Microservices are well suited to be implemented in Docker containers, where each container hosts a service. A computer runs multiple containers at the same time, which makes it easy to simulate complex micro-service architecture.

In view of the characteristics of power information system, based on Docker technology, the typical application scenarios of power information system are divided into one or more micro-services according to their business functions, thus realizing a complete single business logic unit, realizing a set of functional logic combination software packages of the same type or tightly coupled single business objectives or business scenarios, and providing software clients with interfaces, which are accessible. Through PC, mobile devices, large screen and other terminal devices to achieve human-computer interaction. Man-machine interface realizes business logic by calling one or more micro-services, and does not directly access the database. Each micro-service is independently developed, compiled, published, deployed and operated, and the application system is invoked through the distributed service bus. Lightweight communication protocols such as REST or RPC are used between micro-services, and embedded middleware is used to run them.

2. Related Work

Because of the problems of high coupling of system functions and low reusability of business modules in power information system designed by traditional monolithic architecture, it is difficult to meet users' needs. Service-oriented architecture (SOA) was proposed immediately. SOA architecture encapsulates the specific business function modules of information system into service forms, and publishes them in the form of services. The coupling between services is low, so the realization of service interface can be achieved. Select a specific technology route according to the needs, and maximize the reuse of services according to the needs, effectively solve the problems of the single architecture [2-4].

The limitation of the SOA architecture is that the information system designed based on the SOA architecture is difficult to adjust the service load flexibly according to the demand under the circumstance of high concurrency and high real-time business data. At the same time, the difference between the usage of computer resources in the peak and trough periods of information system business makes the operation and deployment of information system more flexible and simple. Microservices has begun to emerge and become the preferred architecture for many distributed systems. Microsoft service architecture can be regarded as an improvement of SOA architecture. By fine-grained partitioning of business functions, Microsoft service architecture ensures the single responsibility of services, and joins service registry, circuit breaker and other modules to meet the performance needs of load balancing system. According to the idea of micro-service architecture, each micro-service runs in an independent process, and services communicate with each other through lightweight communication mechanism. To meet this design requirement, container technology represented by Docker technology has become an important means in the implementation of micro-
service architecture. Zhang Shifeng and others [1] discussed the application progress of Docker technology in micro-services, and summarized the application scenarios of Docker technology. Zhang Yi [7] proposed a lightweight PaaS platform for power grid based on Docker. The platform has good effect in dynamic scaling and efficient migration and deployment of resources. It can improve the efficiency of power system operation and maintenance and resource utilization.

Research has shown that micro-service architecture is the application system architecture chosen by many large cloud platforms. Container technology represented by Docker has been widely used in micro-services. The combination of Docker technology and micro-service architecture greatly improves the performance and efficiency of information systems [5-6, 8].

3. Microservice Architecture

3.1. Overall framework

With the continuous development of distributed generation, controllable load, incremental distribution network and material services, the requirement of frequency, complexity and timeliness of information interaction among the main bodies in power information system is getting higher and higher. The centralized single architecture needs to be restructured to meet the trust needs of multi-participated business systems such as load-storage interaction of source network and precise supply of materials (see Fig. 1).

![Micro system architecture](image)

**Figure. 1** micro system architecture

Power information system micro-service architecture is mainly composed of unified management of distributed service bus and various power system micro-services. The distributed service bus consists of micro-service gateway, service registry, micro-service monitoring center, micro-service configuration center, security control center and other components.

The micro-service gateway provides a unified access entry for front-end applications, and realizes dynamic routing of target micro-services through routing strategy; the registry provides storage of service-related information and calls between micro-services through service registration and service discovery, thus realizing decoupling between micro-services; the micro-service monitoring center monitors and guarantees the running status and invocation links of services. Micro-service runs in a healthy state; micro-service configuration center provides storage of unified dynamic configuration information in multiple environments for micro-service, and synchronizes updated information into micro-service by push-pull mode in each running environment; micro-service serves stateless in a distributed environment, and stores status information into micro-service according to actual business conditions. In the external cache.
3.2. Service registration and discovery mechanism

Power information system micro services are registered and cancelled through service registry, which maintains the service registries of Netflix Eureka and ZooKeeper. All power information system microservices are registered in the service registry. The registry is deployed as a proxy service, runs in the server or virtual machine where the microservice is located, and runs as a container through Docker, as shown in Figure 2.

According to the needs of different business application scenarios, power information system micro-service uses a registration discovery mechanism combining client discovery and server discovery. The client discovery mechanism uses Netflix Eureka technology, and the client queries all available service instances from a service registry service center. The client uses the load balancing algorithm to select one of the available service instances and then sends out the request. The registration management and query of service instances are all connected by calling REST API provided by Eureka in the application. The client of Eureka adopts self-registration mode. The client needs to handle the registration and cancellation of service instances and send heartbeat.

The service discovery mode is used in power information system application scenarios with weak client capability, such as those widely used in power Internet of Things applications. In this mode, the client sends a request to a special load balancer, and the load balancer sends a request to the service registry, forwarding the request to the service instance available in the registry. Service instances are also registered and cancelled in the registry. Load balancing can use Haproxy or Nginx. In this paper, a Docker-based Zookeeper scheme is adopted to realize service-side discovery. ZooKeeper is a mature service discovery system. It was first used in Hadoop and adopted Paxos algorithm. It has the advantages of strong consistency and high availability.

The power information system micro-service monitoring system realizes the decoupling of service discovery component and micro-service architecture by monitoring service and tracking heartbeat to find available service instances and registering and canceling them to the registry.

3.3. Communication mechanism

In monolithic applications, calls between modules are implemented by programming language-level methods or functions. Distributed applications based on micro-services run on multiple machines, and each service instance is a process. Interaction between services must be achieved through interprocess communication (IPC).

The communication between micro services in power information system includes one-to-one interaction and one-to-many interaction, including request response mode, notification mode, and asynchronous response mode, publish-subscribe mode, etc. Power information system micro-service is a combination of various IPC mechanisms. For example, TEST API based on HTTP protocol is used to synchronize micro-services, which can provide lightweight resource access. Asynchronous communication mechanism based on message, such as AMQP or STOMP, can loosely couple dependencies seen by micro-services.
Message queue is the main form of asynchronous communication in power information system micro-service. The scenario of using message queues in micro-service architecture is generally for decoupling between micro-services. There is no need to know which service instance consumes or publishes messages between services. Just deal with the logic of your domain scope, and then publish it through the message channel, or subscribe to the messages you care about. At present, there are many open source message queuing technologies. For example, Apache Kafka, RabbitMQ, Apache ActiveMQ and RicketMQ of Alibaba have now become one of the Apache projects. The message queue consists of the following three components:

- **Producer**: Produce the message and write it to the channel.
- **Message Broker**: Message broker, which manages messages written to channel by queue structure. Responsible for storing/forwarding messages. Broker is generally a cluster that needs to be built and configured separately and must be highly available.
- **Consumer**: the consumer of news. At present, most message queues guarantee that messages are consumed at least once.

### 3.4. Service Gateway

Service gateway provides a unified access interface for power information system applications by using uniformly defined request forwarding and merging, establishes system boundaries, and avoids the intrusion of domain models between services. Service gateway is a Springboot application deployed and run in Docker container like micro-service. As shown in the figure.

![Gateway Deployment Diagram](image)

**Figure 3** gateway deployment

The information of all requests in power information business is aggregated by service gateway, which is the entrance of all client requests. By providing Restful-style API, the only way to access the system is realized. Service gateway mainly solves the problem of information aggregation and client adaptation. By requesting the resource of service gateway API, it can change the upper business without affecting the underlying architecture of micro services. In the cluster environment, the service gateway API can expose unified ports, and its instances will run on different IP servers to achieve load balancing effect. The service gateway can also perform some fault tolerance, security and service degradation for requests of micro services. Using response programming to implement service gateway can make the synchronous and concurrent code of threads more concise and easier to maintain.

### 3.5. Container layout

Power information system micro-service system involves a large number of micro-services. Each micro-service corresponds to one or more Docker containers. Container layout technology is used to manage Docker containers. To deploy power information system micro-service with Docker, it is necessary to package micro-service into Docker image. In order to manage the pull/push image uniformly, it is necessary to build a power industry image library.
After the Docker image is built, because each container runs different instances of microservices, the containers are also isolated from each other to deploy services. By arranging technology, DevOps can lightweight the deployment and monitoring of container management to improve the efficiency of container management.

At present, some common choreography tools, such as Docker Compose, Ansible, Chef, Puppet, can also do container choreography. Because the above tools are common platforms, it is necessary to write scripts according to power business and Docker command to realize specific functions in the application of power information system. In this study, Kubernetes is used to schedule and arrange micro-service containers of power information system. Kubernetes is an open source container cluster management system of Google. It is implemented in Google language. It provides application deployment, maintenance, extension mechanism and other functions. In the cluster environment of microservices, Kubernetes can easily manage instances of microservice containers across machines.

Based on Kubernetes, this research realizes the automatic deployment and replication of Docker micro-service instances in power information system micro-service according to business requirements and load conditions. Micro-service runs in a cluster mode. Integrated management and control interface can manage containers across machines, as well as rolling upgrade and storage arrangement.

Through the built-in Docker-based service discovery and load balancing module, the failure self-repair mechanism can be realized, and the collapsed container can be replaced without interrupting the business. It can be expanded and scaled at any time to increase the flexibility of system management. The minimum management element of micro-service in power information system based on Kubernetes is Pod, each Pod contains one or more containers associated with the business, which is generated or released according to the business status and scheduling situation.

Docker-based power information system micro-service achieves the goal of automated operation and maintenance. In the automated operation and maintenance platform, the micro-service deployed by Docker virtualization technology can be configured once and deployed many times. It can migrate rapidly according to the change of demand. At the peak of business, it can meet the demand by adding containers. At the low end of business, it can reuse computer resources smoothly through container management, which greatly improves the efficiency of operation and maintenance. Automated
operation and maintenance platform uses DevOps (Development and Operations) development mode to realize the life cycle management of application system, enhances the coordination of IT development, operation and quality assurance, breaks the original boundaries between developers and operators, and upgrades the power information system with a smooth curve.

3.6. Load balancing and performance optimization

The strategy of Kubernetes elastic scaling is response scaling according to the load situation. There is a long time delay from load pressure to balanced response. This study predicts the Pod load, studies the characteristics of load changes, and drives Kubernetes to expand ahead of load increase, so as to achieve the purpose of timely response to business needs.

Let \( \{L_t\} \) as load sequence of history time, the exponential smoothing results of the sequence with \( \alpha \) parameter are as follows:

\[
S_t^{(1)} = \alpha l_t + (1 - \alpha)S_{t-1}^{(1)}. \tag{1}
\]

On this basis, the second smoothing results are as follows:

\[
S_t^{(2)} = \alpha S_t^{(1)} + (1 - \alpha)S_{t-1}^{(2)}. \tag{2}
\]

So the prediction equation is as follows:

\[
L_{t,T} = a_t + b_t \Delta T, \tag{3}
\]

Where,

\[
a_t = 2S_t^{(1)} - S_t^{(2)}, \tag{4}
\]

\[
b_t = \frac{\alpha}{1 - \alpha} (S_t^{(1)} - S_t^{(2)}). \tag{5}
\]

The system expands by judging the size of the current value and the predicted value. If the current value is less than the predicted value, it indicates that the load will increase in the next cycle, and the current replica set needs to be expanded. Conversely, if the current value is greater than the predicted value, the resources occupied by the current replica set need to be released. The addition and reduction of replica sets are created and deleted by executing Pod instructions.

4. Simulation analysis

The unified deployment of power business is realized by simulating the micro-service of "National Network Cloud" platform in the laboratory environment. The deployment units such as micro-service and micro-application are containerized, and the deployment of each unit is accomplished through container arrangement, load balancing, elastic scaling and other components, thus realizing the characteristics of high availability, high performance and high concurrency of business applications. Five servers with Intel E5 CPU and 128G memory are used to run CentOS operating system and basic Docker running environment. One server serves as cluster manager and the other four servers serve as computing nodes.
Fig. 5 shows the variation curve of request response time in different experiments. It can be seen from the graph that the delay is less than 2 seconds, which is within acceptable range for most power application scenarios. Figure 6 shows the CPU load curve in different experiments. It can be seen that the CPU load has a surplus of more than 40%, which can ensure the overall performance of power information microsystems.

5. Conclusion
In the energy Internet ecosystem, computer tasks are highly parallelized, and complex concurrent components bring about high failure rate and low system availability. This paper proposes a micro-service scheme based on Docker technology, which can reduce the contradiction between computing speed and data mobility rate by subdividing the granularity of computing resources, thus improving the overall performance of power information system and enhancing user experience.

6. Acknowledgments
This work was financially supported by the science and technology projects of headquarters of State Grid Corporation (Research and application of non-stop state Adaptive upgrade technology for information system)
7. References

[1] Zhang Shifeng, Pan Shanliang. Application of Docker technology in micro-service [J]. Electronic technology and software engineering, 2019 (04): 164.

[2] Fan Lei. Architectural design of enterprise-level micro-service based on Docker [J]. Computer knowledge and technology, 2018, 14 (34): 25-26.

[3] Guo Yanghu. Research and Implementation of Docker Container Scheduling Strategy in Micro-Service Environment [D]. Beijing University of Posts and Telecommunications, 2018.

[4] Wu Jiechu. Research and implementation of micro-service management technology based on Docker-Swarm [D]. South China University of Technology, 2018.

[5] Wang Fangxu. Design of Microsoft Service Architecture Based on Spring Cloud and Docker [J]. China Informatization, 2018 (03): 53-55.

[6] Wu Kun'an, Huang Wensi, Han Zehua, Huang Pingfa, Li Yun, Wang Rongye and Chen Weilun. Design and implementation of database micro-service system based on Docker [J]. Foreign electronic measurement technology, 2017, 36 (12): 57-62.

[7] Zhang Yi, Hu Yonghua, Huang Ding. Construction scheme of lightweight PaaS platform for power grid based on Docker [J]. Information and Computer (theoretical version), 2017 (11): 75-78.

[8] Zhang Shifeng, Pan Shanliang. Application of Docker technology in micro-service [J]. Electronic technology and software engineering, 2019 (04): 164.