Software Architecture Analysis of Intelligent Distribution and Transformation Terminal Based on Container Technology

Yanrong Jiang¹, Shejiao Hu¹, Zhao Niu¹ and Lin Wu²
¹Department of Computer and information, Hefei university of technology, 485 Danxia road, Shushan district, Hefei city, Anhui province, China.
²Anhui Wanhong Electric company limited by shares, 39 Jincheng road, Linquan county, Fuyang city, Anhui province, China.
Email: 2808571445@qq.com

Abstract. Ubiquitous power Internet of things is an important direction of the development of smart grid, and distribution Internet of things is the application of ubiquitous power Internet of things in the field of distribution. As the terminal unit of state perception, intelligent decision-making and control subject in the distribution Internet of things, intelligent distribution and transformation terminal is the foundation to realize the distribution Internet of things. In order to solve the problems of complex software architecture of traditional distribution and transformation terminal, which is not conducive to expansion, maintenance and transplantation, Microservice + container technology is adopted as the software architecture of intelligent distribution and transformation terminal, which reduces the complexity of development, improves the efficiency of operation and maintenance, and meets the diverse and rapidly changing service requirements of distribution network.

1. Introduction

The rapid development of Internet technology and new energy technology has promoted a new round of energy revolution. The combination of smart grid and “Internet +” technology has become a hot topic in the field of electrical engineering. Therefore, the ubiquitous power Internet of things has sprung up, which has expanded more open development space for the construction of smart grid.

The distribution Internet of things is a new type of power network operation form generated by the integration of traditional industrial technology and Internet of things technology. Through the extensive interconnection, intercommunication and interoperability between equipment, it realizes the comprehensive perception, data fusion and intelligent application of the distribution network. As the core equipment in the automation construction of low-voltage distribution network, the performance of intelligent distribution and transformation terminal will directly affect the monitoring and management level of distribution network.

The main control module of intelligent distribution and transformation terminal is designed according to the principle of hardware platform, which can be freely assembled and flexibly expanded according to the rapidly changing service requirements. After the support of hardware platform, software definition is the core of intelligent distribution and transformation terminal. By using the way of software-defined terminal, the distribution and transformation terminal is changed from traditional equipment to platform, which realizes the deep decoupling between hardware resources and software application, and adds and deletes business functions at any time without changing hardware.

However, with more and more business modules connected to distribution and transformation terminals, if the traditional single architecture mode is adopted, the software complexity will...
inevitably increase exponentially, and if any part of it is changed, the entire application needs to be redeployed, so agile development and rapid repair are impossible. Microservice architecture is a direction of the development of software architecture towards flexible dynamic scaling and distributed architecture, and it fits the design concept of software-defined terminal.

2. Microservice Architecture + Container Technology
As a new programming architecture model, the core idea of microservice is to divide a complex and complete application program into many independent and small services. Lightweight communication mechanism is used to communicate and cooperate with each other among services to realize the functions of the complete application program.

2.1. Main Advantages of Microservice Architecture
- Each component service has an independent running process, which can be deployed independently, independent of other service modules, to achieve decoupling between services.
- In the microservice architecture, each component service only focuses on a single service. When building each component service, you can choose the right language and tools.
- When a component service fails, the failure will be isolated in that component service, which will not make the whole application unavailable, thus reducing the risk and improving the fault tolerance rate.
- Each component service has a single function, so the business is clear and the amount of code is small, which makes maintenance and migration easy.

Although microservices have many advantages, the management of a large number of microservices is a big problem. In addition, if microservices are developed in different programming languages, the actual deployment needs to load their own libraries, thus increasing the complexity of deployment. The emergence of container technology solves these problems perfectly. Container technology reduces the implementation cost of microservices, makes microservices move easily between different platforms, and provides the foundation and guarantee for the large-scale use of microservices.

2.2. Container Technology
The traditional virtual machine technology is to divide the hardware into different logical partitions in the form of software, and install Guest OS on the logical partition to achieve isolation and availability. Every virtual machine needs to run a complete operating system and a large number of installed programs. The portability of applications is also very cumbersome. Compared with the traditional virtual machine technology, the container technology does not simulate the hardware operation. It is a kind of kernel lightweight operating system layer virtualization technology. Based on Linux container, the implementation is guaranteed by Layer DFS, Namespace and Control group. The underlying technology supporting container technology is shown in figure 1.

![Figure 1. Implementation of docker container](image-url)
Layer DFS mainly implements hierarchical file storage, which is the basis of building container image. Namespace is the default API provided by the Linux operating system, which realizes the isolation between containers, so that each container has its own independent resources. Control group is also a function provided by Linux operating system, which provides a means to manage memory resources, so that system administrators can appropriately limit the consumption of container resources.

2.2.1. Characteristics of container technology. The container packages the relevant program code, function library and environment configuration files required by each application to form a container image. Without installing the operating system and building the environment, developers can develop and test based on the container image. The emergence of container technology makes people pay more attention to applications, ignore operating systems and dependent environments. The container technology has the following characteristics:

- Each container only packs the necessary Bin / Lib, and each host can run hundreds or even thousands of containers at the same time.
- Each container is an independent and complete execution environment, independent of the external environment.
- "One time encapsulation, running everywhere", each container can fast migration according to the change of demand.
- Containers can complete the creation and destruction of applications in seconds.
- All containers on the same host share the same operating system kernel, which leads to less waste of resources.
- Developers only need to pay attention to the programs running in the container, and operation and maintenance personnel only need to pay attention to the management of the container, so as to improve the work efficiency.

3. Function Analysis of Intelligent Distribution and Transformation Terminal

In order to meet the basic requirements of ubiquitous power Internet of things, such as comprehensive state perception, efficient information processing, convenient and flexible application, the low-voltage distribution network data acquisition and automation system should be developed into a cloud platform-edge equipment architecture. The Client/Server mode is adopted for communication between intelligent distribution and transformation terminal and cloud platform. The former is the client side and the latter is the server side, and the selected communication protocol is Modbus-RTU. The communication architecture of low-voltage distribution network is shown in figure 2.

![Communication architecture of low voltage distribution network](image-url)
After fully investigated the development status and problems of distribution transformer terminal, Arm + Linux is used as the system architecture of intelligent distribution and transformation terminal, and a variety of communication interfaces are expanded to realize the intelligent perception, monitoring and management of other auxiliary equipment in the station area. Terminals can exchange real-time information with the master station, provide operation status, equipment status and other auxiliary information of the distribution network, and meet the design requirements of the new generation of distribution network for intelligent distribution and transformation terminals.

3.1. Software Requirement Analysis

In order to effectively solve the problems of scattered equipment installation, single function and repeated installation in the distribution station area, and improve the data processing ability and information interaction ability of the terminal, the intelligent distribution and transformation terminals must have the following functions:

- Monitor the operation of the station area, including monitor the parameters of the power grid, monitor the temperature and humidity, monitor the status of remote signaling switch, etc. Analyze the collected electric energy information, send the control instruction to the intelligent capacitor when the parameter exceeds the limit, execute the corresponding conditional capacitor switching strategy to realize the local reactive power compensation, and report the abnormal record to the master station.
- Own edge computing technology, when the users have abnormal power consumption, the fault data can be analyzed locally and processed by intelligent calculation, and then sent to the master station for comprehensive judgment.
- Send the meter reading command to the intelligent electricity meter in the station area, read the electric energy data in the intelligent electricity meter, and complete the monthly electricity consumption statistics as the basis of electricity consumption charges.
- GPRS and Ethernet communication modes can be used to communicate with the master station. Terminals can upload the operation information of the station area to the master station and receive various control commands from the master station at the same time.
- Use MySQL as the data storage container to store the power grid operation parameters, terminal operation abnormal events and power consumption information.
- Create the system daemons, which are responsible for starting the functional processes and circularly detecting the running status of the functional processes.

4. Realization of Software Architecture of Intelligent Distribution and Transformation Terminal

The software architecture of intelligent distribution and transformation terminal based on container technology includes operating system layer and application program layer. Linux operating system kernel adopts version 4.15, which provides many hardware device drivers and API, and enables namespace, control group, container engine, etc. to ensure the realization of docker function. The application layer consists of a large number of containers, and different programs run in different containers. The software architecture is shown in the figure 3.
4.1. Construction and Deployment of Containers

In order to realize the complex functions of intelligent distribution and transformation terminals, it is necessary to create multiple containers. The process of deploying multiple containers is very tedious, for example, to maintain multiple images, manage the creation, start, stop, delete of multiple containers, etc.

The deployment method is to divide the containers on the same host into five groups according to the functional requirements, which are monitoring the operation status of the station area, storing data, remote control of the master station, reading the power consumption information in the smart meter, system daemons, then use docker compose to manage the containers of each group uniformly. The following are the specific implementation steps:

- First, install harbor to build a private image warehouse, then use five groups of source code to generate executable files, use Dockerfile files to build images, and finally, push images to image warehouse. When new application services need to be deployed, they can be regenerated only by loading the images that have been packaged and published in the image warehouse.
- Define the containers needed by each group through docker-compose.yml file, such as specifying the creation method of the container, the network connected by the container, the port mapping relationship, the environment variables required by the container, etc.
- Then through the command can create or manage multiple containers according to the definition of the docker-compose.yml file, which greatly reduces the workload.

4.2. Limit Resources used by Containers

The system resources of intelligent distribution terminal are limited, such as CPU, memory and so on. By default, there is no resource restriction on the container. Container can use all the resources scheduled by the kernel, which will eventually lead to the lack of system resources, and kernel will calls OOME to kill some processes to release resources. The stability of the system will be seriously affected.

When starting the container, in addition to limiting the maximum memory occupied by the container through commands, you can also set the priority of the container through CPU shares, and set the priority of the important containers to occupy as much CPU resources as possible, so as to improve the resource utilization.

Set the priority of container for system daemon to the highest. Containers for monitoring station area parameters and reading smart meter information will interact with I/O devices., so the priority of this part of containers is set higher. The container for storing data only needs to complete execution without emphasizing response speed, so its priority is set to the lowest.

4.3. Communication Between Containers

The communication between containers is an important part of microservice architecture. Network
namespace realizes the network isolation between containers, which makes it impossible for each container to access each other in the traditional way.

Containers deployed on the same host can communicate with each other in many ways. After evaluation, the user-defined bridge mode is selected for communication. When creating a container, specify the container to connect to the user-defined bridge, so that the container does not need to know the IP address of other containers and the user-defined bridge provides automatic DNS resolution between containers, so that containers can communicate with each other directly through the container name. The communication mode is shown in figure 4.

![Figure 4. Communication mode between containers](image)

### 4.4. Persistent Data Storage

The data generated in the container is only temporarily saved in the container. If the container is deleted, the data will also be lost. The database container of the intelligent distribution and transformation terminal is used to store the operation parameters of the power grid in the station area, the abnormal operation events of the terminal and the power consumption information in the station area. If the database container is deleted by mistake, the result is unacceptable.

The scheme of persistent data storage adopted by intelligent distribution and transformation terminal is Bind Mounting. When creating a MySql database container, the /var/lib/MySql directory of the specified container is attached to a new directory of the host, and the data generated when the container runs will be written to the directory of the host. The two directories are synchronized. If the files in the host directory are modified, the files in the container directory will also be modified. If the container is deleted, the database file can also be found in the mount directory of the host computer, thus realizing the persistent storage of data.

### 4.5. Collection and Exhibition of Logs

In order to ensure the normal operation of the container and the service quality of the intelligent distribution and transformation terminal, the collection and display of logs is a problem that must be solved. Log is a very important means to acquire the running state of the containers. It can display error, info and even debug information. The logs generated by the container are stored in their own containers. So when the container exits, the generated logs will be deleted, which brings some challenges to the collection of logs.

ELK stack technology is used to collect the logs of large-scale containers for storage and display. The specific process is to use Logstash to collect the logs in each container, store the collected logs in Elasticsearch, and visually display the log data through Kibana. ELK stack has the characteristics of high expansibility, easy to use and high query efficiency. It is convenient for developers to debug applications and locate problems by checking logs in real time.

### 5. Experiment and Result Analysis

After the requirement analysis of intelligent distribution and transformation terminal, the design and implementation of software architecture and the design of cloud platform management system, the
system joint commissioning is carried out.

5.1. Experimental Environment
The experimental environment is shown in Figure 5.

![Experimental environment](image)

**Figure 5.** Experimental environment

The parameter configuration of the host is shown in table 1.

| Name               | Version and model                          |
|--------------------|---------------------------------------------|
| Docker             | 19.03.5-ce                                  |
| Operating system   | Ubuntu 16.04                                |
| Harbor             | Harbor-offline-installer-v1.5.0.tgz         |
| Docker-compose     | 1.24.1                                      |
| Kernel             | 4.15.0                                      |
| RAM                | 512M                                        |
| Memory             | 4G                                          |
| Processor          | ARM Cortex-A8                               |

5.2. Test Result
Some test results are shown in the figures.
Figure 6. Display the data collected by distribution transformer terminal on cloud platform

![Data Display](image)

Figure 7. Display part of the data in a curvilinear way

The results show that the operation of intelligent distribution and transformation terminal is stable and meets the expected demand.

6. Conclusions

Based on the research background of smart grid construction, this paper investigates the functional requirements and development direction of intelligent distribution and transformation terminal. According to the guiding ideology of software definition terminal, adopts the microservice + container technology as software architecture, and the standard development environment is built through images, so as to realize the unified deployment and management of the application in the intelligent distribution and transformation terminal, and improve the application scalability and portability.

Through the concept of container as a service can rapidly expand functions, shorten application
development cycle, improve product iteration speed, more meet the basic needs of the "extensive interconnection, comprehensive perception, hierarchical intelligence" of the ubiquitous power Internet of things, more fit the development trend of smart grid construction and changeable business needs.

7. References
[1] Licheng L, Yongjun Z and Zexing C. 2016 J. Aoeps. 40 pp 1-9
[2] Licheng L, Yongjun Z and Min X. 2017 J. De. 2 pp 1-9
[3] Dongsheng Y, Daohao W and Bowen Z. 2019 J. Pgt. 40 pp 107-14
[4] Xun Z, Chengyu L and Jinwu W. 2018 J. Ts. 34 pp 162-70
[5] Siyao L, Qiang L and Bin L. 2015 J. Ceas. 36 pp 110-13
[6] Ting Y, Feng Z and Yingjie Z. 2019 J. Aoeps. 43 pp 9-20
[7] Jun L, Wenpeng L and Riliang L. 2018 J. Psy. 42 pp 3108-15
[8] Jun L, Wanxing S and Riliang L. 2019 J. Hve. 45 pp 1681-88
[9] Min X.2017 J. Aoic. 34 pp 62-6
[10] Chengshan W, Fengzhang L and Tianyu Z. 2016 J. Hve. 42 pp 2017-27
[11] Xiangjun D, Jinli W and Dezhi F. 2017 J. Pst. 41 pp 2709-15
[12] Jichuan Z, Lei C and Mingyu Z. 2019 J. Hve. 45 pp 1-8
[13] Yue M and Gang H. 2015 J. S. 36 pp 10-4
[14] Bo D, Xue W and Fei S. 2016 J. Johnu. 43 pp 327-30
[15] Kai W, Gongxuan Z and Xiumin Z. 2015 J. Ctad. 25 pp 138-41
[16] Yao X and Zhixiang Z. 2019 J. Cade. 47 112919
[17] Jun L, Wenpeng L, Riliang L, Peng W and Jianying L. 2018 J. Pst. 42 10
[18] Riliang L, Haitao L, Shengfeng X, Dong Y, Yan W and Lihan C. 2019 J. Hve. 45 pp 1707-17