**DESIGN AND IMPLEMENTATION OF POWER BIG DATA PLATFORM**

**ABSTRACT**

In recent years, with the advancement of the national construction of smart grid and the development of power grid enterprise integration system, the traditional power data platform has some defect, such as insufficient scalability, repeated implementation of two sets of logic of off-line data warehouse and real-time data warehouse, highly correlated storage of data warehouse during computation, and difficulty in platform migration, which restrict the deeper mining and analysis of power big data. Therefore, this paper adopts a technical architecture incorporating distributed calculating, micro-service, flow-batch integration and lake-warehouse integration, combining the current popular Web front-end and back-end technologies such as Vue, SpringCloud and Flask, and utilizing big data components such as Hadoop, Flink, Hudi and Kafka as well as Docker containerization technology. We explore and build a power grid system operation oriented big data platform that assembles real-time computing, multi-source heterogeneous data storage, and develop APIs for authority management, information management, data analysis and visualization, machine learning and other data supporting services that assist power enterprises in managing electrical equipment and users' power consumption.With the help of this platform, power enterprises can intelligently track the equipment status, view the offline and real-time visual charts to analyze the power consumption by users, so as to have better basis in the power dispatching, maintenance and scheme adjustment of electrical market.

**CHAPTER ONE**

**Introduction**

Power data has the characteristics of multi- source, heterogeneity, large volume, rapid growth and low rate of utilization. The data can be roughly divided into three categories: monitoring data of power grid operation, marketing data and management data of power enterprise, which can be derived from smart meters, sensors and other devices as well as multi- party information platforms of power grid operation parts such as power generation, transmission and energy consumption. Various types of power data differ greatly in data structure, including structured data represented by tabular data recording equipment properties, environmental parameters and other information, unstructured data represented by video, audio, pictures and documents generated by monitoring system, and semi structured data represented by interface type data in JSON or XML format from other business databases. However, in the actual production practice, there is often a phenomenon of massive data but lack of information which can be effectively excavated. The use of the latest data processing tools and the design of an analysis applications based on big data technology can effectively improve the computational efficiency and comprehensive analysis ability of power grid data, provides more valuable information for the subsequent real- time decision- making and management, and thus brings new development opportunities for the construction of smart power grid. In the early 21st century, Google published three technical papers introducing extensible distributed file system GFS [1], programming model MapReduce for parallel analysis of large-scale datasets [2], and distributed storage system BigTable for massive data [3]. These three systems were later transformed into Hadoop's core architecture, namely the distributed file system HDFS, distributed computing framework MapReduce, and distributed database HBase. Today, Hadoop has become an indispensable component in many enterprise big data architectures. Its distributed computing and storage capabilities offer high reliability, scalability, efficiency, fault tolerance, and low cost, making itself widely applicable in resource management, job scheduling, data storage, data analysis, and other domains, therefore can meet the requirements of electric power data platforms.

In practical production scenarios, data with the same metric may need to be generated in real-time by streaming tasks as well as offline by batch tasks. Reusing the results from stream processing in batch processing can effectively reduce the workload for developers. However, in real-time scenarios, due to the issue of data disorder, the data produced by stream processing may deviate from the data obtained from batch processing, resulting in compromised data quality. The first-generation distributed open-source stream processing engines, represented by Storm, sacrificed result accuracy for lower latency but couldn't guarantee "exactly-once" consistency. The second-generation stream processors, with the Lambda architecture as the core, combined the first-generation stream processors with traditional batch processors to ensure both low latency and high accuracy,but was had difficulty in establishment and maintenance. As a representative of the third-generation stream processors, Flink, with the Kappa architecture as its core, not only inherits the merit of the second but also features high throughput, high availability, and effectively enables the reuse of stream processing results during batch processing. Boshra Pishgoo [4] and others proposed a Hybrid Distributed Batch-Stream (HDBS) architecture for real-time data anomaly detection, demonstrating that this architecture can guarantee accuracy like batch processing while maintaining quickness like stream processing.

In recent years, the Lakehouse [5] has emerged as a new data management architecture, which combines the structure and management capability of data warehouse with the low-cost storage and flexibility of ‘data lake’. With the underlying storage in widely adopted data formats and the complex mechanisms of upper metadata layer aiming at transaction management, version control, and SQL operations, this architecture can overcome challenges such as inconsistent Lakehouse data, invalid data caused by ETL delays, weakness in complex analysis, and expensive costs, it can further provides a consistent interface for higher-level services such as business intelligence, reporting analysis, data science, and machine learning.

The ability to handle large-scale and high-concurrency response is also an assessment criterion for the platform. In the early days of the internet, for pursuit of simple implementation , most applications adopted monolithic architectures, where all business functionalities were developed, packaged, and deployed within a single project. But drawbacks including high coupling between functional modules impede maintenance and further development makes it incapable of handling high-concurrency access and storing massive amounts of data. In 2014, Martin Fowler [6] introduced the concept of microservice, effectively addressing these issues by adopting a single responsibility principle of breaking down application services into individuals developed and deployed respectively.

Based on these findings, we design and build a large power data platform utilizing big data frameworks such as Hadoop, Flink and Hudi, combined with Web application technology frameworks such as Vue [7] and SpringCloud [8], and adopt the technical architecture of lake and warehouse integrated storage, micro-service, flow and batch integrated computation. Finally, a prototype system is developed based on the above platform, which realizes the management operation of user authority, data storage and update, dynamic data visualization and intelligent detection.

**2. Architecture Design and Technology Selection**

*2.1. Overall architecture*

This platform can be divided into three components: big data module, back-end, and front-end. The overall architecture is shown in Figure 1.



**Figure 1.** Overall architecture of the power big data platform

*2.2. Big Data Module*

The main tasks of the platform layer are, first, to provide a distributed, highly error tolerant, scalable file system for storing massive multi-source heterogeneous data; second, to do ETL, offline computing, and real -time computing for the historical data and daily incremental data stored by the platform to support dimensional modeling and indicator system construction, thus meeting the demand for data statistics in the backend part. For the former task, we built distributed Hadoop clusters through Docker containerization, while the latter relies on Flink based, stream and batch integrated architecture and Hudi based, lake warehouse integrated architecture in implementation.

Containers and clusters. To shield the environmental dependencies of each software on the physical machine it belongs to and facilitate platform migration, all sub projects of this platform are deployed in the Docker container [9], and all big data components are installed in the Docker container. Having considered the availability and fault tolerance of the big data platform, as well as the development stage of this project, this platform temporarily set three Docker containers as three nodes of the cluster, naming the host ‘master’, the others ‘slave1’ and ‘slave2’, following a ‘one master, two slaves’ mode. Additionally, give the adoption of micro-service architecture by the backend , the system can meet the current project requirements in terms of performance, availability, scalability, and scalability.

Data warehouse and data lake. The data sources of this platform are mainly divided into three categories:

1. Relational business data, such as user data, device basic data, operation records, etc. They are stored in MySQL [10] as relational tables and can be imported into web pages.
2. Image files, model files, and CSV files on HDFS [11]. The images describing device diagram and defect diagram are stored in HDFS in the format of ".jpg", while the machine learning models are stored in HDFS in the format of ". model". The mapping relationships of all images, models, and related information are maintained through CSV files.
3. Real time data simulated by Mock. This data simulates the real-time data sent by various power devices with embedded systems to the designated port of this platform, and it is sent to Kafka's designated topic in the form of JSON.
4. Data belongs to the first type should be imported into the data lake by FlinkCDC through monitoring MySQL binlog. Data belongs to the latter two types would be first transmitted to the specific topic of Kafka for necessary peak clipping, and then sent to the ODS layer of the data lake.The arriving data in the data warehouse would undergo the following five layers of streaming data processing [12]:
5. ODS (Operational Data Store) layer, also known as the raw data layer, retains the raw data collected directly from the business system.
6. DIM (Dimension) layer, namely the public dimension layer, is built based on dimensional modeling and used to store dimension tables of the dimension model.
7. DWD (Data Warehouse Detail) layer, that is, the detail data layer, is built based on dimensional modeling to store the fact table in the dimensional model.
8. DWS (Data Warehouse Summary) layer, also known as the summary data layer, is built based on an indicator system to store summary tables with common statistical granularity.
9. ADS (Application Data Store) layer, also known as the data application layer, is used to store the results of various statistical indicators.

The workflow of loading the original data in three types is as follows: structured data like arrays of data needed by online computing should be firstly stored in Zookeeper node and then transported in a pipeline manner to Kafka queue which docks with the entry of ODS layer, ones like historical statistics record would undergo aggregation and computation operations by MapReduce, then with the help of Yarn's job scheduling functionality, be stored in HDFS together with unstructured data which represented by ‘.csv’, ‘.pdf’, etc, finally extracted and carried by data collector called ‘flume’to Kafka. Semi-structured data including database tables, as reference for various data operations, should be imported into data warehouse and stored permanently, but without the need for previous preprocessing tools.

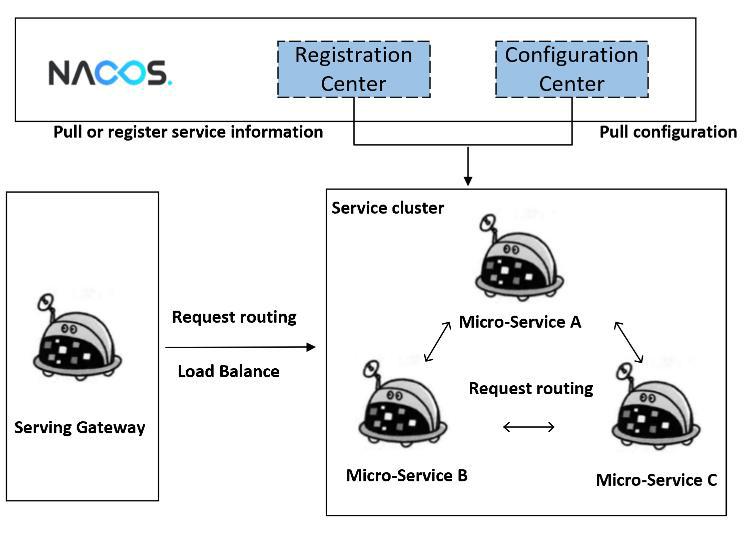
*2.3. Backend*

The backend of the system consists of a service layer and a storage layer, with the main function of processing and forwarding front-end data requests to specific microservice, and utilizing the storage and computing capabilities of the big data platform to meet various business requirements.

Service layer. The backend module is the main provider of data services, and its core is to provide various microservice. This module mainly consists of five SpringCloud microservice and one Flask microservice. The former is a collection of Java microservice that process the vast majority of business data requests in the system, while the latter is a Python micro-service registered in the form of microservice in the Nacos service registry to integrate Python based machine learning modules.

The main technical frameworks used for the backend module are SpringCloud and MyBatisPlus, supplemented by Flask. SpringCloud is a popular Java language web backend development framework that provides developers with a convenient tool to quickly build common patterns of backend application services. MyBatisPlus is an excellent persistence layer framework that provides convenient API for adding, deleting, modifying, and querying standard SQL databases [13]. Flask is a flexible and easy-to-use Python language web backend development framework that facilitates the integration of machine learning modules in this platform [14].

The management of various microservice relies on various microservice components, including service registry Nacos [15], remote service invocation OpenFeign, and business gateway Gateway, those components transform a application task into a collection of small, independent services working in parallel, thus enable scalability, flexibility, convenience in development a nd maintenance. The microservice design scheme is shown in f igure 2 .



**Figure 2.** Microservice architecture design solution

Storage Layer. The data storage of various microservice relies on MySQL, Redis, Nacos, and HDFS. As a relational database, MySQL is used to store the most important business data of the system. The data is logically stored in the form of an E-R relational graph model and interacts with microservice through ORM mapping, while physically stored in the form of files on the hard drive.

Redis, as a non relational memory database, is used for MySQL data caching. After microservice startup, it loads some frequently accessed data into memory in advance to accelerate the processing efficiency of server-side data requests [16].

The configuration center of Nacos centrally manages the configuration files of various microservice and supports hot updates of configurations.

HDFS, as a distributed file system, is used to store massive unstructured data of the system (such as device defect maps), as well as various historical and real-time data, forming the foundation of streaming data access in OLAP scenarios. The streaming data lake platform Hudi also stores data based on HDFS.

*2.4. Frontend*

The front-end module is the access point for system users. Platform users generally access the front-end module through a browser, log in to the system as a specific user in the front-end module, and then interact with the platform. The main function of the front-end module is to provide the platform with pages for data display, image display, and business operations, while also being able to perform routing interception to ensure that users with specific roles access specific pages.

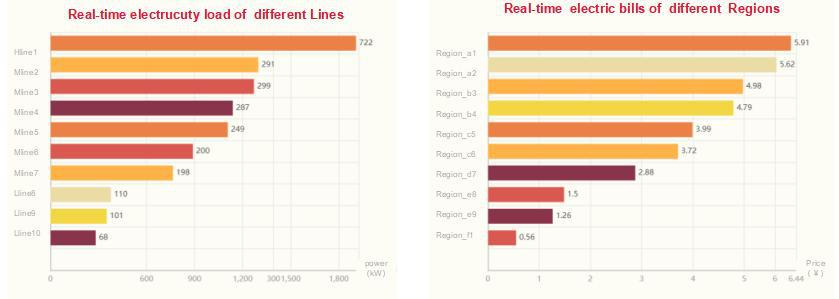
The front -end module uses the Vue3 framework, Vite scaffolding[17], ArcoDesign component library [18], and Echarts chart library [19] as technical support, and sends data requests to the back-end module through Axios. Vue3 is a popular Javascript UI framework that provides easy-to-use APIs and learning documents, and has an excellent fully responsive rendering system. Using Vite to build front-end projects can achieve rapid cold start and real-time hot module updates, greatly improving the development experience. ArcoDesign, as the middle and back end front-end solution launched by ByteDance, provides a set of front-end components based on the design principles of timely feedback, close to reality, system consistency, error prevention, and compliance with habits, greatly simplifying the development of front-end components.

**3. Demonstration of prototype implementation**

The prototype system of the proposed power big data platform is implemented by leveraging various technology artefacts. The core functions are real- time data processing, import and storage of large batch of files, and intelligent anomaly detection on equipment images.

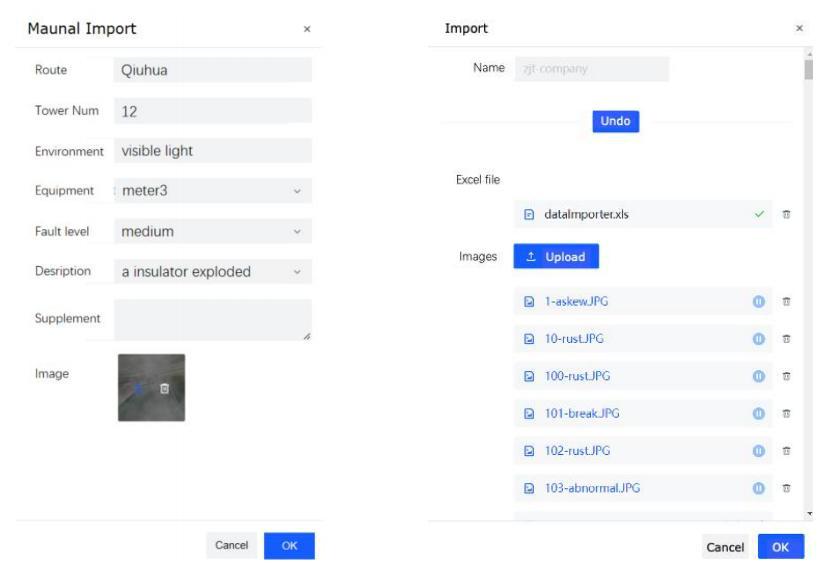
*3.1. real-time data processing*

We simulated real-time data by adding noise to public sequential dateset recording time series data with 1 minute interval, the consumption of each device accumulates or updates in the same interval. The screenshot of real-time electrical consumption visualization interface is displayed in figure 3.



**Figure 3.** Simulated real-time data visualization *3 . 2 . Batch data processing*

Batch processing is enabled by importing thousands of batches containing equipment images and tabulate containing the matching descriptions within few minutes, also, the screenshot of that module is as shown in figure 4.



**Figure 4.** Interface of images import

*3.3. Anomaly detection*

Anomaly detection, as an advanced application of the stored images, can be realized based on any pre- trained image classification model, such as VGG [2 0 ] , ResNet [2 1 ] , Vision Transformer ( ‘ ViT’ in abbreviation) [2 2 ] and Swin Transformer( ‘ SwinT’ in abbreviation) [2 3 ] .

Demonstratively, we have built three types of datasets containing images of electrical equipment from real scenario, named ‘ MIDDLE’ and ‘ LARGE’ , the latter is obtained via data augmentation on ‘MIDDLE’ . The detail of the three datasets is shown in t able 1 . To test the performance of each aforementioned model, we split each dataset into train, valid and test set with ratio of 8:1:1, which are used to train and evaluate the models.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Sample** | **Positive** | **Negative** |
|  |  |  |  |
| **MIDDLE** | 1892 | 273 | 1619 |
| **LARGE** | 3257 | 1638 | 1619 |

**Table 1.** information of the test dataset

Since the task is factually a binary classification, the basic evaluation metric is Accuracy ‘ACC’,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| whose formulaic definition is equation (1). |  | TP+TN |  |  |  |
|  |  |  | (1) |  |
|  | ACC = ~~TP+TN+FP+FN~~ | | |  |  |

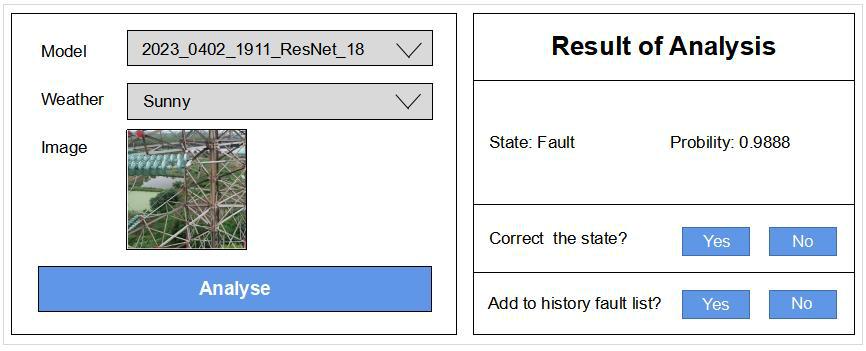
TP, FP, TN, FN are refer to true positive, false positive, true negative and false negative, respectively. Considering the distribution between two types of samples is imbalanced in ‘MIDDLE’, we additionally introduce AUC and AUPRC, the former one reflects sensitivity of a model and the latter reflects precision and recall rate of the model based on all thresholds. The commonality of the two is that the higher the value, the better the classification ability of the model.

The results listed in table 2 below are from experiments in our previous work [2 4 ] . Among the tested models, ResNet and ViT perform well on both balanced and imbalanced dataset, moreover, given the mean gap of 4.27% between the two models on ‘LARGE’ dataset and relatively minus gap of - 1.47% on ‘MIDDLE’, we choose to implement ResNet into the detection module.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **LARGE** | |  | **MIDDLE** | |
| **Methods** | ACC | AUC | AUPRC | ACC | AUC | AUPRC | |
| VGG11 | 0.497 | 0.513 | 0.512 | 0.81 | 0.604 | **0.246** | |
| ResNet 18 | **0.761** | **0.844** | **0.826** | 0.81 | 0.61 | 0.216 | |
| ViT | 0.706 | 0.789 | 0.808 | **0.818** | **0.624** | 0.238 | |
| SwinT | 0.61 | 0.627 | 0.661 | 0.812 | 0.522 | 0.159 | |

**Table 2.** Test results of the models on anomaly detection

As shown in figure 5 , for an unlabeled image, the detection module can quickly output whether the equipment suffers malfunction with probability of 98.8%.



**Figure 5.** Prediction of an unlabeled image

**4. Discussion**

This article starts with the problems existing in traditional big data platforms, builds a distributed platform based on virtualization containers, combines dimensional modeling and big data technologies, integrates real- time electrical data calculating and visualization as well as machine learning model to simulate the important part of power data processing occasion. By adopting the architecture concept of integrating flow and batch, using Flink to build a real- time data warehouse, our platform avoids the problem of duplicate implementation of offline and real-time code logic in traditional data warehouses, keeps a good balance between the accuracy and efficiency of real- time calculations; based on the architecture integrating lake and warehouse, Hudi for data management, the platform also solves the problem of high correlation between traditional data warehouse calculation and storage, and inconsistency between lake and warehouse data.

This platform has been able to effectively provide data management and analysis services for power enterprises about the information of electricity consumption and equipment status. Next, the LDAP protocol can be used for centralized management and maintenance. By introducing the LDAP protocol for centralized management and maintenance of account information, the problem of cross system identity authentication can be avoided.

**Acknowledgement**

The work is under the auspices of State Grid Zhejiang Electric Power Co., Ltd., science and technology project (5211HZ220004).

**References**

1. S Ghemawat and H Gobioff and S T Leung. The Google file system[J].*ACM SIGOPS Operating Systems Review*, 2003,37(5):29-43.
2. J Dean and S Ghemawat. MapReduce: simplified data processing on large clusters.[J]. *Commun.ACM*,2008,51(1):107-113.
3. F Chang et al."Bigtable". Operating systems design and implementation. pp.205-218.
4. B Pishgoo and A A Azirani and B Raahemi. A hybrid distributed batch-stream processing approach for anomaly detection[J]. *Information Sciences*,2021,543:309-327.
5. M Zaharia, Ali Ghodsi, Reynold Xin, Michael Armbrust: Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics.*Proceedings of CIDR*.2021
6. T J. Microservices[J]. *IEEE software*, 2015, 32( 1): 116- 116.
7. Vue.js. (2023).Vue.js. https://cn.vuejs.org.
8. Spring. (2023). Spring Cloud. https://spring.io/projects/spring-cloud.
9. Docker. (2023). Get started with Docker. https://docs.docker.com/get-started/overview
10. DuBois P. MySQL[M]. *Pearson Education*, 2008.
11. Borthakur D. HDFS architecture guide[J]. *Hadoop apache project*, 2008, 53( 1- 13): 2.
12. Chakravarthy S, Jiang Q. Stream data processing: a quality of service perspective: modeling, scheduling, load shedding, and [complex event processing[M].](https://flask.palletsprojects.com/en/2.2.x/) *[Springer](https://flask.palletsprojects.com/en/2.2.x/)* Science *& Business Media*, 2009.
13. MyBatisPlus. (2023). MyBatisPlus official website. https://baomidou.com.
14. Flask. (2023). Flask official [website.](https://cn.vitejs.dev/guide/why.html.) https://flask.palletsprojects.com/en/2.2.x/
15. Nacos. (2023). What is Nacos?. https://nacos.io/zh[-cn/docs/what-is-nacos.html.](https://arco.design/docs/spec/philosophy.)
16. Redis.(2023). https://redis.com.
17. Vite. (2023). Why Vite?. https://cn.vitejs.dev/guide/why.html.
18. ArcoDesign. (2023). Design Philosophy. https://arco.design/docs/spec/philosophy.
19. Apache Echarts.(2023).Apache ECharts official website. https://echarts.apache.org/zh/index.html
20. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv*:1409. 1556 (2014).
21. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
22. Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv*:2010. 11929 (2020).
23. Liu, Ze, et al. "Swin transformer: Hierarchical vision transformer using shifted windows." *Proceedings of the IEEE/CVF international conference on computer vision*. 2021.
24. Zhao, Zhao, et al. "Multi-Modal Cross-Attention Learning on Detecting Anomalies of Electrical

Equipment." *International Conference on Pattern Recognition, Machine Vision and Intelligent Algorithms*. 2022