Eng Int Syst (2026) 1: 27–38

Engineering
Intelligent Systems

Experimental Detection Data Collection Based on Intelligent Access Adaptation

Shu Yang[1,a*], **Ziqiang Zhou**[1,b], **Wei Gao**[2,c], **Qiang Xue**[1,d] **and Jiani Zhang**[1,e]

[1] *State Grid Shanxi Electric Power Co., LTD, Electric Power Science Research Institute, Taiyuan 030001, Shanxi, China*
[2] *State Grid Shanxi Electric Power Co., LTD, Taiyuan 030021, Shanxi, China*

The real-time collection and processing of experimental and testing data are crucial for ensuring product quality and promoting technological innovation in the fields of industry and scientific research. However, traditional data collection methods suffer from issues such as insufficient standardization, inconsistent interfaces, and delayed data processing, which limit the effective utilization of data. This study investigated a method for collecting experimental detection data based on intelligent access adaptation, aiming to address the aforementioned challenges. The research involved the design of a RESTful API (Representative State Transfer Application Programming Interface) and a unified interface in JSON (JavaScript Object Notation) format, and the construction of a real-time data stream processing platform based on Apache Kafka. ElasticSearch and Pandas libraries were used to intelligently adjust data formats, and gRPC (Google Remote Procedure Call) protocol and Protobuf (Protocol Buffer) data format was used to optimize data transmission efficiency. In regard to the effectiveness of the proposed method, the experimental group reduced data collection latency by about 28.7% compared to traditional methods, demonstrated by the reduction in the page loading time from 738 milliseconds to 526 milliseconds, and a performance score of 92 points, significantly higher than the control group's 68 points. In terms of classification accuracy, the experimental group achieved an accuracy rate of over 96% in five data acquisition scenarios: electricity, meteorology, magnetic field, distance, and velocity. The recognition precision of the electricity category reached 0.988. The testing of interface compatibility showed that the experimental group exhibited total compatibility, while the control group had multiple compatibility deficiencies. The system throughput monitoring shows that the throughput of the experimental group is closer to the target value, indicating greater processing efficiency.

Keywords: intelligent access; data collection; real-time processing; experimental testing; Internet of Things; data adaptation

1. INTRODUCTION

In modern industry and scientific research, the collection and processing of test data is crucial to ensuring product quality, improving production efficiency and promoting technological innovation. Traditional data collection methods are unable to cope with the rapid development of intelligence and real-time data acquisition [1]. Such methods usually rely on manual operation and batch processing, lack unified standards and interfaces resulting in cumbersome data collection processes and long delays, and are unable to achieve real-time monitoring and facilitate rapid decision-making [2], thus affecting the accuracy and effectiveness of the data. Therefore, addressing the problem of efficient, accurate and real-time data collection and processing is a matter of urgency [3]. With the development of emerging technologies such as the Internet of Things, artificial intelligence and big data analysis, the field of data collection and processing has ushered in new opportunities. These technologies have greatly improved the automation of data collection, reduced the need for manual intervention, and improved the real-time capture and accuracy of data [4]. However, it is still challenging to combine these technologies effectively in

*Address for correspondence:Shu Yang, State Grid Shanxi Electric Power Company Electric Power Science Research Institute, Taiyuan 030001, Shanxi, China, Email: 3948445985@qq.com, zhou8085@126.com, gaowei0121@126.com, xueqiang12098773@yeah.net, h335978970@163.com

order to build an efficient test data collection system. The problem with traditional methods is that the data standards and interfaces are not unified, making it difficult for devices to exchange data [5]. Different devices and systems often use different data formats and communication protocols, resulting in poor interoperability and making data integration and analysis complicated [6–7]. The traditional batch processing mode also causes delays and cannot meet the demand of modern industry for real-time data [8]. The problems of data compatibility and accuracy also need to be solved urgently. The inconsistency of equipment standards leads to errors and losses during transmission and processing. The development of Internet of Things technology connects sensors and detection equipment to achieve automatic data collection and transmission, thereby improving efficiency and accuracy [9]. However, how to achieve seamless connection and efficient data transmission between different devices is an issue that still needs to be resolved [10]. Most existing research focuses on the single link between data collection and transmission, lacking a comprehensive analysis of the entire system. The advantages of artificial intelligence technology in data processing and analysis provide a new means for solving these problems. The use of machine learning and deep learning can produce intelligent data analysis and improve data accuracy and reliability [11]. However, it is still difficult to effectively integrate artificial intelligence into the data collection system to form a complete intelligent data process. The rapid development of big data technology has provided strong support for data storage, processing and analysis, and improved the efficiency of data utilization [12]. However, how to effectively apply big data technology to data collection systems to achieve real-time processing and analysis still requires in-depth research.

This paper conducts research on test detection data collection methods based on intelligent access adaptation, aiming to solve the problems of inconsistent standards, incompatible interfaces and data delays in traditional methods. By designing a unified interface in RESTful API and JSON format, a real-time data processing platform based on Apache Kafka and Flink is constructed to achieve efficient real-time data collection. At the same time, the ElasticSearch and Pandas libraries are used to ensure the compatibility of data formats, and gRPC and Protobuf are used to optimize data transmission efficiency. Experimental results show that this method significantly improves the efficiency and accuracy of data collection, providing a reliable solution for industry and scientific research fields.

2. RELATED WORK

In the field of modern testing, the efficiency and accuracy of data collection are crucial. Traditional methods often suffer from insufficient standardization, long delays, and limited processing capabilities [13]. In order to deal with these problems, intelligent access adaptation technology is being applied. RESTful API and JSON format are widely recommended as standards for seamless data connection. RESTful API shows significant advantages in terms of data exchange due to its simplicity and flexibility [14]. JSON has become the preferred format for data transmission due to its lightweight and easy-to-read characteristics. Using OpenAPI specifications for interface design helps ensure standardization, and generates documentation and test cases through the Swagger tool to support developers to quickly verify the correctness of the interface [15]. Alshraiedeh explored in depth the RESTful architectural style in his research, laid a theoretical foundation for interface design, and significantly improved the interoperability between different systems [16]. Frozza analyzed the advantages of JSON as a data exchange format, demonstrating its efficiency and simplicity in data transmission. Real-time data collection occupies an important position in modern experimental testing [17]. Traditional data collection methods often rely on batch processing, resulting in delays and difficulty in meeting the needs of instant decision-making. The data stream processing system based on Apache Kafka provides an effective solution. As a high-throughput distributed messaging system, Apache Kafka supports real-time data stream processing. Combined with Apache Flink, it can further enhance the real-time capture and continuity of data [18]. Hassan described in detail the architecture of Apache Kafka and its advantages in processing large-scale real-time data streams [19]. Kekevi discussed the integration of Apache Flink and Apache Kafka, pointing out that this combination can significantly reduce data processing latency and improve real-time performance [20]. These studies provide theoretical foundation and practical experience for achieving real-time data collection.

In intelligent access adaptation, optimizing the adaptation layer is the key to ensuring data compatibility and accuracy. Intelligent data adaptation algorithms, such as ElasticSearch's data format adaptation, can achieve automatic adjustment of data formats and effectively solve the problem of the inconsistent data formats of different devices and systems. As an efficient search engine, ElasticSearch can handle real-time queries of large-scale data and support the conversion of multiple data formats [21]. Combining data cleaning and conversion with the Pandas library in Python can further improve data quality and accuracy. Kathare's research promoted the application of ElasticSearch in large-scale data processing [22], while Saabith discussed in detail the data processing function of the Pandas library, providing practical tools and methods for data cleaning and conversion [23]. These studies provide theoretical and technical references for the optimization of the adaptation layer. Data processing and transmission are crucial in test and detection data acquisition systems. The optimization of data processing and transmission protocols, and the adoption of gRPC and Protobuf data formats can help reduce data transmission delays [24]. As an efficient remote procedure call protocol, gRPC provides excellent communication capabilities. Combined with Protocol Buffers for data serialization, it can significantly improve the transmission rate and efficiency [25]. Karcher's research introduced the design concepts and application scenarios of gRPC and Protocol Buffers, emphasizing their importance in efficient data transmission [26]. Studies have shown that the combination of gRPC and Protobuf can significantly improve data processing speed and efficiency. After achieving the intelligent access

Figure 1 Real time acquisition process.

adaptation test data acquisition system, it is important to evaluate the system's impact in order to determine its performance [27].

3. RESEARCH METHODS

3.1 Real-time Data Collection

This study used Apache Kafka as the data streaming platform and integrated it with Apache Flink in order to perform real-time data processing in the experimental testing data collection system, so as to achieve real-time data collection and reduce data latency. This method can solve the problems of excessive data collection delay and the inability of standard batch processing mode to meet real-time decision-making requirements. Figure 1 shows the entire real-time collection process.

The use of high-precision sensors for the real-time monitoring of necessary data is the initial stage of the data collection process. These sensors can capture various environmental, chemical, and physical characteristics and convert the information into digital signals. Sensors can use HTTP (Hypertext Transfer Protocol) to send data to the collection terminal. Transfer data to the collection terminal to ensure consistency and accuracy of data transmission. The acquisition terminal can handle the conversion and preprocessing of various data formats, and is equipped with data interfaces tailored for different sensor types. It also provides the necessary drivers and middleware to ensure interoperability with various data sources.

This article chooses Apache Kafka as the real-time data streaming technology to meet the requirements of high throughput and low latency data transmission. Kafka can efficiently process large amounts of data to meet experimental needs, thanks to its distributed, scalable, and highly available architecture [28]. The configuration process includes creating themes to classify and store different types of sensor data, and configuring producers and consumers to ensure that data can be sent and received in a timely manner. When deploying a Kafka cluster, it is necessary to configure and tune the Broker nodes to ensure stable performance even in high concurrency situations. Throughout the entire process, sensors collect data in real-time and send it to Kafka producers. Kafka producers send sensor data to Kafka clusters, which receive the data and forward it to Kafka consumers and Flink tasks.

Apache Flink, as a distributed stream processing framework, performs real-time data processing and has powerful real-time data processing capabilities. The specific implementation steps include configuring Flink tasks to consume data from Kafka topics, and writing Flink jobs to implement real-time processing logic for data, such as data cleaning, filtering, aggregation, and anomaly detection. By deploying Flink clusters, efficient processing of streaming data can be achieved, and processing logic and resource allocation can be dynamically adjusted according to business needs. The processed data is sent back to the Kafka cluster for Kafka consumers to use, and Kafka consumers can transfer the processed data to the data processing and storage system for storage, for subsequent analysis and use.

3.2 Data Transmission and Processing

In the experimental detection data acquisition system based on intelligent access adaptation, the data processing and transmission link is a key part to ensure the efficient operation of the system. To reduce latency during data transmission, this article adopts efficient gRPC protocol and Protocol Buffers data format.

Performance metrics are crucial when choosing the appropriate data transmission protocol. There are significant differences in latency, throughput, resource consumption, and other aspects among different protocols. For the

Table 1 Performance comparison of different transmission protocols.

Performance Metric	gRPC	REST	WebSocket	MQTT	AMQP
Average Latency	5 ms	100 ms	20 ms	10-100 ms	5–20 ms
Throughput (msg/sec)	10,000	1,000	5,000	1,000	2,000
Connection Establishment Time	$\sim$ 100 ms	$\sim$ 200 ms	$\sim$ 150 ms	$\sim$ 200 ms	$\sim$ 300 ms
Message Size Limit	4 MB (default)	No explicit limit (depends on server)	No explicit limit (depends on implementation)	256 MB (configurable)	No explicit limit (depends on broker)
Resource Consumption	Low (optimized for performance)	Moderate (depends on server load)	Low (persistent connection)	Low (lightweight protocol)	Moderate to High (depends on implementation)
Error Rate (%)	$< 1\%$	1–5%	$< 1\%$	$< 2\%$	$< 1\%$
Payload Compression	Supports gzip compression	Supports gzip compression	Supports gzip compression	Supports payload compression	Supports payload compression

purpose of comparison, this article conducted a detailed performance data comparison of various commonly used data transfer protocols by collecting various materials, including gRPC, REST (Representational State Transfer), WebSocket, MQTT (Message Queuing Telemetry Transport), and AMQP (Advanced Message Queuing Protocol). Table 1 shows the performance of each protocol on key performance indicators, which helps to analyze the advantages and disadvantages of each protocol in depth and make informed choices [29–30].

gRPC performs well on several key performance indicators, making it the preferred data transfer protocol. Its average latency is less than 5 milliseconds, compared with 100 milliseconds, 20 milliseconds, 10–100 milliseconds and 5–20 milliseconds for REST, WebSocket, MQTT and AMQP respectively. This advantage is particularly useful in online games and financial transactions. This is especially noticeable in real-time applications. gRPC can process about 10,000 messages per second, which is significantly higher than REST's 1,000, WebSocket's 5,000, MQTT's 1,000, and AMQP's 2,000. It is suitable for high-concurrency and data-intensive applications, such as IoT data processing and large-scale microservice architecture. In terms of connection establishment time, gRPC is about 100 milliseconds, compared with 200 milliseconds for REST, 150 milliseconds for WebSocket, 200 milliseconds for MQTT and 300 milliseconds for AMQP, showing its lower interaction latency. gRPC consumes less resources when processing a large number of requests, ensuring good scalability of the system. Compared with REST and AMQP, gRPC consumes less resources under high load, and its low error rate provides reliability of data transmission in critical business scenarios. gRPC supports two-way streaming, allowing the client and server to send and receive messages at the same time, which is suitable for real-time communication and data updates. However, REST and MQTT have limited capabilities in two-way communication. Although AMQP supports asynchronous messaging, it is not suitable for real-time interaction. Using Protocol Buffers for data serialization, gRPC ensures the strong typing of the data structure and reduces the risk of errors during development. Compared to REST, WebSocket, and AMQP, which may encounter type mismatch issues when dealing with complex data structures, it reduces potential errors and maintenance costs. With its superior performance in latency, throughput, connection time, resource consumption, reliability, bidirectional flow and strong data typing, gRPC has become an ideal choice for modern microservice architecture and real-time applications, meeting the performance and reliability requirements of complex systems.

After adopting gRPC protocol as the main technology for data processing and transmission, this study further improved the performance and efficiency of the system by adding protocol buffers. Protobuf, which is a language-neutral and platform-independent binary serialization format, can effectively define data structures and generate corresponding code through compilers to ensure compatibility of data across different systems. GRPC utilizes Protobuf to achieve efficient data serialization and deserialization by defining service interfaces and data message types in the. proto file, ensuring fast response and low latency when transmitting data over the network [31]. This combination enables gRPC to optimize bandwidth usage and improve data processing speed when building high-performance distributed systems.

The first implementation step involves setting up a development environment to ensure the normal operation of gRPC and Protobuf libraries. Developers can initially create a pro file to define the format of data messages and service interfaces. These files provide detailed descriptions of the service's methods, data structures for requests and responses. Subsequently, these files are converted into code in the target programming language using the Protobuf compiler, and the generated code can be used for subsequent server-side and client-side implementations.

Afterwards, the corresponding server logic needs to be implemented. Developers need to create a gRPC server and register the service to ensure that it can process requests sent

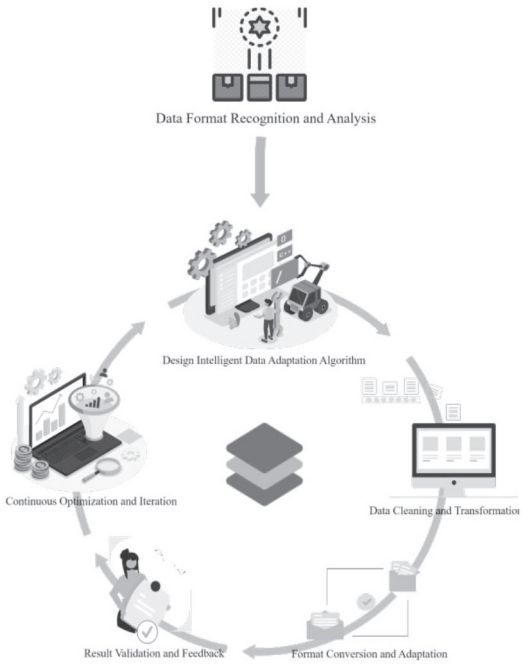

Figure 2 Adaptation layer optimization steps.

by clients and return corresponding results. At the same time, the client implements data interaction by calling the server's interface, sending requests and receiving responses. After the implementation is completed, comprehensive functional testing can be conducted to determine whether the communication between the server and client is smooth, and to ensure that the system can run as expected. After deployment, it is necessary to regularly maintain and update the system, expand according to business needs, adjust the proto file to adapt to new functional requirements, and regenerate corresponding code to maintain the flexibility and efficiency of the system [32].

3.3 Adaptation Layer Optimization

After completing the various data transmission steps, processing, and real-time data collection, this study further optimized the adaptation layer to ensure that the system can handle various data formats and maintain efficiency and accuracy during transmission. In order to optimize the adaptation layer, this study selected ElasticSearch and Pandas libraries. ElasticSearch is a distributed search and analysis engine that can quickly process and analyze large amounts of data. The Pandas library is a powerful data processing and analysis tool that facilitates data cleaning and transformation [33]. Below is a detailed introduction to the optimization process of the intelligent data adaptation algorithm. The overall steps are depicted in simplified form in Figure 2.

ElasticSearch is used in the initial stage of collection to index and examine information collected from various devices. The goal of this stage is to search for data formats, fields, and their types. ElasticSearch allows the user to import data from sensors and other devices, as well as create matching indexes for effective data storage and retrieval. After completing the index, ElasticSearch's query and aggregation methods can be used to conduct in-depth research on data field types, data distribution, and missing values.

In this study, an intelligent data adaptation method was chosen for the optimization of the adaptation layer. After the data format is identified, it dynamically matches and converts data of various formats, ensuring the accuracy and flexibility of the data in the entire adaptation process. Based on the results of data analysis, the guidelines and strategies for data conversion can be clarified. The scripting and query language functions of ElasticSearch can be used to create an automatic data translation method. In order to ensure the accuracy and flexibility of data adaptation, this study selected an algorithm based on the dynamic modification of the conversion technology of the incoming data attributes. Due to its dynamic adjustment ability, the system can respond quickly to different data formats and demand changes [34].

Once the algorithm design has been completed, the Pandas library can be used for data conversion and cleaning to improve the quality of the data. Then ElasticSearch can be used for format conversion to ensure compatibility, and an appropriate conversion method can be selected based on intelligent adaptive algorithms. Following the conversion of the data format, the results are validated to ensure accuracy, and any issues that arise during the entire adaptation process are recorded and checked to determine the areas that need to be developed [35]. As adaptive layer optimization is a continuous process, it is crucial to regularly collect input from users and system functions, identify problems, and improve adaptive algorithms.

3.4 Intelligent Interface Design

After completing real-time data collection, data processing and transmission, and adaptation layer optimization, the next step is intelligent interface design. Efficient interface design can ensure the smooth integration of data between various devices and systems, thereby improving the collaboration capability and data flow efficiency of the entire system [36].

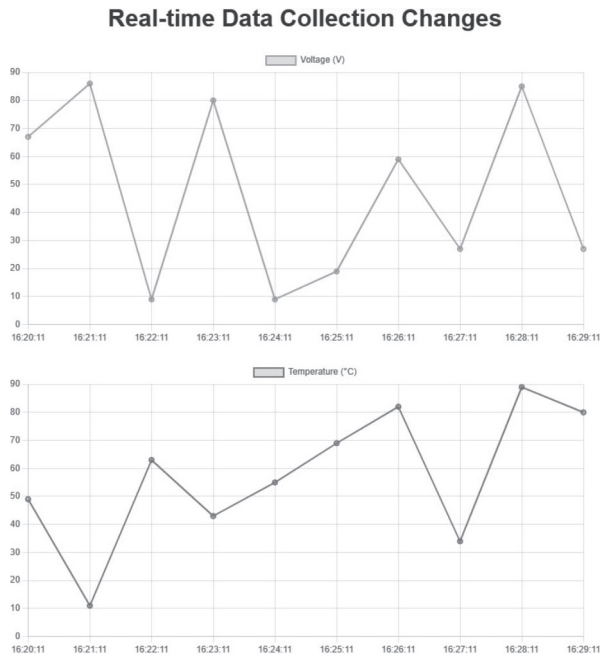

Figure 3 Partial data display obtained from the interface.

This section explains in detail the specific process and technology involved in intelligent interface design, including the use of Swagger tools and OpenAPI standards to build test cases and interface documents, as well as the use of RESTful API and the JSON format.

In order to determine the precise requirements for data interaction, a requirements analysis must be performed for each module and device in the system before starting the interface design. The goal of this phase is to determine the functions that the interface must provide and the direction of data flow. Based on the analysis results, a single standard RESTful API interface can be created to ensure seamless data exchange between different components. The endpoints of the API (such as /pai/v1/devices for device management and /pai/v1/data for uploading and downloading data) can be found throughout the process. At the same time, in order to ensure compliance with the RESTful style and achieve an effective interface design, relevant HTTP methods are provided for each endpoint, including GET for data query, POST for data creation, PUT for data update, and DELETE for data deletion [37]. This process is closely linked to the aforementioned data processing and transmission links, ensuring the seamless flow of data at the interface level. Choosing JSON as the data exchange format is to ensure lightweight and easy parsing of data transmission. Specifically, the structure of the request and response bodies for each API endpoint can be defined to ensure consistent data format, and the functions and types of each field can be described in detail in the interface document. For different requests, the specific data type of each field can be specified and the required fields can be marked.

After the interface design has been completed, supporting documents are essential. This study adopted the OpenAPI specification to write interface documentation, ensuring the clarity of each API endpoint, request, and response format. At the same time, the Swagger tool can be used to automatically generate API documentation for developers to easily view and use. Next, test cases can be generated for each API endpoint to verify the usability and stability of the interface, ensuring normal interaction between modules. To maintain backward compatibility of the interface, a version management mechanism can be established by adding version numbers to the API path for subsequent updates and maintenance. Finally, a monitoring mechanism can be implemented to monitor the real-time usage and performance of the interface, collect indicator data through integrated monitoring tools, regularly analyze interface logs, identify potential issues, and collect feedback on usage. By following these steps, the efficiency and reliability of interface design can be ensured, providing strong support for the continuous optimization of the system. In order to better monitor and collect data, this study was able to obtain corresponding data images by calling these APIs through web pages, as shown in Figure 3.

4. EFFECT EVALUATION

To demonstrate the effectiveness of the method proposed in this article, a relatively traditional experimental data collection method was selected as the control group. The control group used dedicated hardware equipment and a centralized data collection system for automated data collection and recording. The data was transmitted wirelessly to a central server and stored in a relational database for management and analysis.

Data collection delay is one of the key indicators when measuring system performance. In order to measure accurately the total time from data generation to system reception, this study used Pingdom for latency testing. Pingdom not only has global monitoring nodes and high-precision monitoring capabilities, but also provides detailed reporting and analysis functions, which are easy to configure and use, making it very suitable for the evaluation requirements of this study.

Firstly, a new monitoring task needs to be created on the Pingdom platform. When configuring monitoring tasks, the

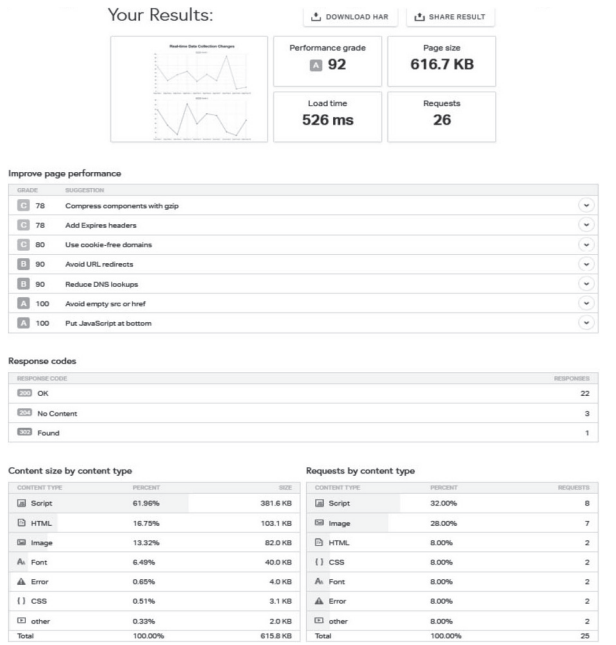

Figure 4 Pingdom report for the experimental group.

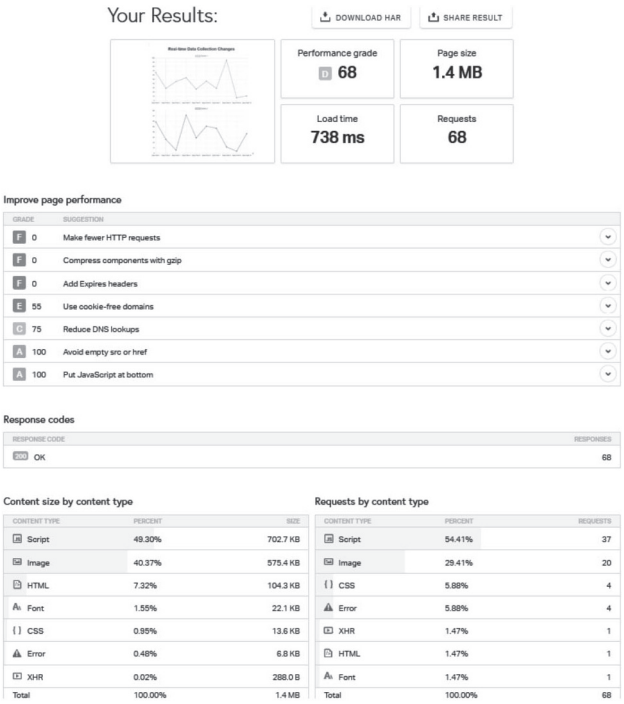

Figure 5 Pingdom report for the control group.

API guided web pages of the experimental and control groups can be set as monitoring objects. To ensure the convenience of the experiment, the monitoring node that can be selected is Tokyo, Asia. Next, this study set the monitoring time interval and alarm threshold. After starting the monitoring task, Pingdom can periodically send requests to the specified API, record the time from sending to receiving a response for each request, and generate detailed delay reports. These reports contain information such as response time, number of failed requests, and their reasons, helping to analyze data collection delays and identify potential performance bottlenecks. After testing, Pingdom provided the monitoring reports shown in Figures 4 and 5.

From the comparison of the two Pingdom reports, it can be seen that the experimental group has a significant advantage in terms of performance. The experimental group scored 92 points for performance, while the control group scored only 68 points. A higher score indicates that the experimental group performs better in terms of loading speed and optimization level. The page loading time of the experimental group was 526 milliseconds, while the control group was 738 milliseconds, which was 28.7% faster.

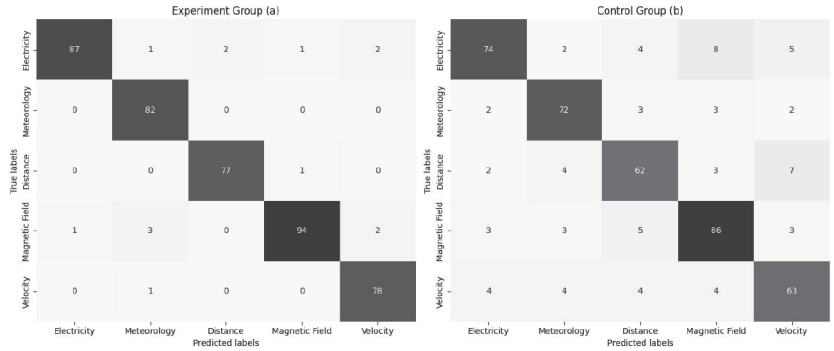

Figure 6 Confusion Matrix for Different Scenarios. Figure 6(a). Experimental group confusion matrix. Figure 6(b). Control group confusion matrix.

Table 2 Evaluation indicators for the experimental group and control group.

	Group	Experiment group	Control group
Accuracy	0.968	0.826	
Precision	Electricity	0.988	0.8706
	Meteorology	0.953	0.8471
	Distance	0.975	0.7955
	Magnetic Field	0.979	0.8269
	Velocity	0.951	0.7875
Recall	Electricity	0.935	0.7968
	Meteorology	1	0.878
	Distance	0.987	0.7955
	Magnetic Field	0.94	0.8608
	Velocity	0.987	0.7975
F1 Score	Electricity	0.961	0.8322
	Meteorology	0.975	0.8623
	Distance	0.981	0.7955
	Magnetic Field	0.959	0.844
	Velocity	0.968	0.7925

The experimental group also outperformed the control group in regard to multiple optimization suggestions. In terms of enabling Gzip compression and adding Expires headers, the experimental group scored higher, indicating that they have more comprehensive data compression and caching strategies. These optimization measures significantly reduce the amount of transmitted data and the number of repeated requests, thereby reducing the latency of API calls. The experimental group scored full marks in avoiding empty src or href attributes and placing JavaScript at the bottom of the page, demonstrating their excellent practices in resource management and loading order, which also helps reduce API call latency and improve overall performance. Through effective resource management and optimization strategies, users' waiting time when using the application has been reduced, thereby improving user satisfaction and application fluency. Overall, the optimization measures taken by the experimental group have resulted in superior performance in practical applications, providing users with more efficient and reliable services.

When it is necessary to simultaneously measure data collected from different environments in the background, whether the system can accurately classify is also one of the accuracy measurements required in this study. This study uses confusion matrices to analyze the classification accuracy of the experimental group and the control group in five different data collection scenarios: electricity, weather, distance, magnetic field, and velocity, comprising a total of 432 data points. This can help visualize the performance differences between two systems when processing these different types of data. Figure 6 shows the confusion matrix of the experimental group and the control group.

In the experimental group, the classification effect was relatively ideal, especially for the power category, with 87 correct classifications and only a few misclassifications. The performance for the meteorological and magnetic field categories is also excellent, correctly classifying 82 and 94 times respectively. The correct classification numbers for distance and speed categories are 77 and 78, respectively, with relatively fewer misclassifications.

In contrast, the classification performance of the control group is slightly inferior. The correct classification of the electricity category is 74 times, and there are more misclassifications. The numbers for the correct classification for the meteorological and magnetic field categories are 72 and 86, respectively, which are slightly lower than those of the experimental group. The correct classification numbers for distance and speed categories are 62 and 63, respectively, with more significant misclassification.

Overall, the classification performance of the experimental group is significantly better than that of the control group, indicating that the experimental group has better performance in terms of classification accuracy.

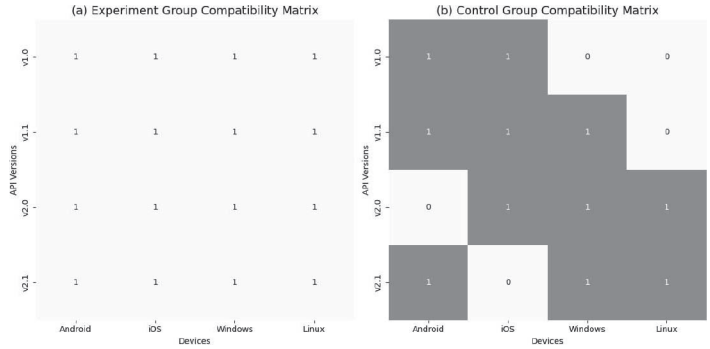

Figure 7 Compatibility Matrix Diagram. Figure 7 (a). Compatibility matrix of experimental group. Figure 7 (b). Compatibility matrix of control group.

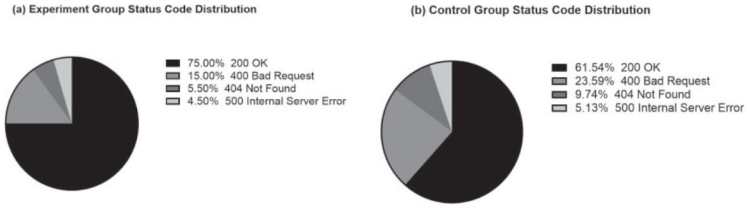

Figure 8 Distribution of Status Codes Obtained from Testing. Figure 8(a). Distribution of Status Codes in the Experimental Group. Figure8(b). Distribution of status codes in the control group

The corresponding evaluation indicators can be further calculated from the confusion matrix. The specific data indicators are shown in Table 2.

Table 2 shows the differences in performance between the experimental group and the control group in terms of accuracy, precision, recall, and F1 score. The overall accuracy of the experimental group is 0.968, significantly higher than the control group's 0.826, indicating that the experimental group has higher accuracy in classification tasks.

In terms of the recognition effect of specific categories, the experimental group has a more obvious advantage. In the field of electricity, the recognition precision of the experimental group reaches 0.988, while that of the control group is 0.8706. In the field of meteorology, the precision of the experimental group is 0.953, while that of the control group is 0.8471. The precision of the experimental group in measuring distance is 0.975, while the control group achieves 0.7955; the precision of the experimental group in detecting magnetic fields is 0.979, and the precision of the control group is 0.8269; the accuracy of the experimental group in measuring speed is 0.951, and the accuracy of the control group is 0.7875. In each test item, the recognition accuracy of the experimental group is generally high, and the experimental group also shows obvious advantages in regard to recall. In power data collection, the recall rate of the experimental group is 0.935, and the recall rate of the control group is 0.7968; in meteorological data collection, the recall rate of the experimental group is perfect, while the score of the control group is only 0.878. In distance measurement, the recall rate of the experimental group is 0.987, and the recall rate of the control group is 0.7955; the recall rate of the experimental group in magnetic field detection is 0.94, and the recall rate of the control group is 0.8608; the recall rate of the experimental group in speed measurement is 0.987, and the recall rate of the control group is 0.7975. The recall rate of the experimental group is significantly higher than that of the control group.

The advantage of the experimental group is further reflected in the F1 score. In the field of electricity, the experimental group's F1 score is 0.961, while the control group's is 0.8322. In the field of meteorology, the experimental group's F1 score is 0.975, while that of the control group is 0.8623. In the field of distance measurement, the experimental group's F1 score is 0.981, while for the control group it is 0.7955. The magnetic field detection experimental group's F1 score is 0.959, and the control group's F1 score is 0.844; the speed measurement experimental group's F1 score is 0.968, and the control group's F1 score is 0.7925. In each test category, the experimental group's F1 score is better than the control group, indicating that the experimental group's performance is more balanced in terms of precision and recall.

Using the Swagger tool to test the interface, this study further determined the compatibility between the experimental group and the control group. Swagger not only provides automatic document generation and API debugging functions, but also ensures seamless connection between multiple devices and systems through its defined interface specifications. By comparing the performance of the interface of the experimental group with that of the control group, the compatibility of different API versions on various devices can be clearly identified, providing an important basis for subsequent system optimization. Figures 7 and 8 show the summary of the test results.

When conducting a thorough analysis of the compatibility and status codes in Figure 8, significant differences in interface compatibility and status code performance between the experimental group and the control group can be clearly identified, demonstrating their respective system performance and user experience. In terms of compatibility, the experimental group performed significantly better than the control group.

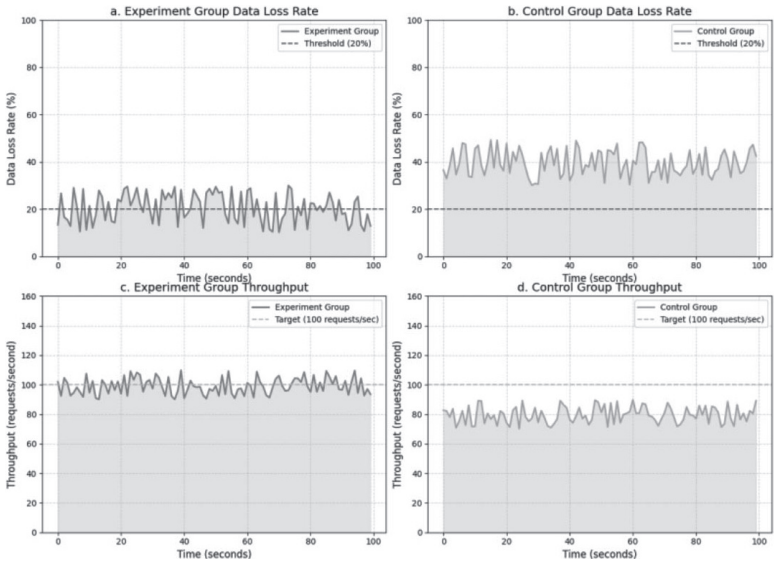

Figure 9 Performance Indicator image. Figure 9(a). Experimental group data loss rate image. Figure 9(b). Control group data loss rate image. Figure 9(c). Experimental group system throughput image. Figure 9(d). Control group system throughput image.

The experimental group demonstrated complete compatibility between all devices and API versions, and all positions marked as "1" in the matrix indicate that the combination of the device and API can seamlessly integrate. This high level of compatibility means that the API implemented by the experimental group on different devices is consistent, capable of effectively handling various requests and returning correct responses.

The area with a value of "0" in the control group compatibility matrix indicates that a specific API version is incompatible with a specific device. This mismatch may make it difficult for users to access certain features or it may cause other problems, thereby reducing user satisfaction and the overall performance of the system. Therefore, in addition to laying a solid foundation for the implementation and use of the system, the compatibility advantage of the experimental group also provides important support for improving user satisfaction.

There is a significant difference in the distribution of status codes between the experimental and control groups. The frequency of "200 OK" status codes was significantly higher in the experimental group than in the control group, indicating that the experimental group processed requests faster and responded more successfully. This high percentage of successful responses indicates that, apart from demonstrating the stability and reliability of experimental group in answering requests, users can have a more seamless and enjoyable experience when using the experimental group's system. Moreover, the reduction of "404 Not Found" and "400 Bad Request" status codes in the experimental group shows a decreased probability of system failure. This change effectively improves the experimental group's practical use, simplifies access to key functions, and reduces the generation of inaccurate answers. In contrast, the control group performed poorly in terms of status codes, particularly the high incidence of "404 Not Found" and "400 Bad Request". This not only increases the problems of request errors and resource not found, but may also have a negative impact on the overall availability and efficiency of the system.

As can be seen from Figure 8, the experimental group performed better in regard to status code and interface compatibility. The research conclusions allow people to better understand the effectiveness and value of the experimental group in actual scenarios. These results explain the effectiveness of the experimental group in compatibility testing and provide solid support for system optimization, user experience improvement and the planning of future developments.

Following the compatibility test, this study used the Grafana tool to track the key performance indicators of the system to achieve an accurate assessment of the system's stability and reliability. The key indicators monitored include data loss rate and system throughput, which can comprehensively map the performance of the system at various time periods. By comparing the monitoring data of the experimental group with that of the control group, this study ascertained the specific differences in the performance of the two groups under different scenarios, and then drew conclusions that have practical application value. Figure 9 shows the data loss rate and system throughput indicators monitored and reported by the Grafana tool. In this chart, the safety threshold of the data loss rate is set at 20%. If this limit is exceeded, it may have an adverse impact on the quality of service. At the same time, the baseline target of the system throughput is set at 100 requests/second, and achieving this goal is critical to meeting business needs.

A comparison of Figures 9a and 9b shows that the experimental group and the control group experienced data loss at different time points. The loss rate of the experimental group is generally lower, usually within the range of 10% to 30%, while the loss rate of the control group fluctuates within the range of 30% to 50%. The significant difference in data loss rate between the two indicates that their data transmission systems are more reliable. Specifically, in the experimental group, the number of data loss rates exceeding the preset standard of 20% is much lower than that in the control group, which further highlights the superiority of the experimental group in data management performance.

Figures 9c and 9d present the system throughput of the two groups. The throughput of the experimental group remained stable at 90 to 110 requests per second, approaching the target value of 100 requests per second, while the throughput of the control group was between 80 and 100 requests per second. Not only is the throughput of the experimental group closer to the ideal state: it also shows better processing efficiency. Conversely, the throughput of the control group failed to reach the target multiple times, indicating the inadequacy of its system performance. Based on these data, it can be concluded that the experimental group showed better performance in terms of both data loss rate and system throughput, making their system more advantageous and efficient in practical applications.

5. CONCLUSIONS

This study focuses on intelligent access adaptation and reconstructs in detail the various aspects of experimental detection data collection. By adopting a unified intelligent interface based on RESTful API and JSON format, this study successfully achieved seamless data integration between various systems and devices, thereby significantly improving data interoperability. At the same time, a data stream processing platform based on Apache Kafka was introduced to achieve real-time data collection and processing, greatly reducing delay in the collection process and ensuring the continuity and real-time nature of the data. This study also used ElasticSearch and Pandas libraries to intelligently adjust the data format, further improving the accuracy and compatibility of the data. Furthermore, the Protobuf data format and gRPC protocol were used to improve transmission efficiency and reduce data transmission delay. After multiple rigorous evaluations, it was confirmed that the proposed solution improved the accuracy, efficiency and real-time performance of data collection. In addition to advancing significant technological development in the industrial and scientific fields, this also points the way for future advancements in data collection technology.

COMPETING INTEREST

The authors declare that no competing interests exist.

FUNDING

State Grid Shanxi Electric Power Company Technology Project: Research on Key Technologies for Intelligent Recording of Electric Power Testing Instruments and Meters (No. 52053024000S).

REFERENCES

1. Roh Y, Heo G, Whang S E. A survey on data collection for machine learning: A big data-ai integration perspective. IEEE Transactions on Knowledge and Data Engineering, 2019, 33(4): 1328–1347.
2. Jentoft N, Olsen T S. Against the flow in data collection: How data triangulation combined with a 'slow' interview technique enriches data. Qualitative Social Work, 2019, 18(2): 179–193.
3. Jain N. Survey versus interviews: Comparing data collection tools for exploratory research. The Qualitative Report, 2021, 26(2): 541–554.
4. Aad G, Abbott B, Abbott D C, Abed Abud A, Abeling K, Abhayasinghe D K, et al. ATLAS data quality operations and performance for 2015–2018 data-taking. Journal of Instrumentation, 2020, 15(04): p04003-p04003.
5. Hariri R H, Fredericks E M, Bowers K M. Uncertainty in big data analytics: survey, opportunities, and challenges. Journal of Big data, 2019, 6(1): 1–16.
6. Zhang B. Real-Time environmental data collection and fusion method for intelligent greenhouses based on agricultural internet of things technology. Engineering Intelligent Systems, 2023, 31(3): 195–204.
7. Han L, Gao L, Feng S, Qu F, Song B. Remote monitoring system based on the internet of things and monitoring methodfor design of construction machinery. Engineering Intelligent Systems, 2025, 33(5): 485–498.
8. Zhan C, Zeng Y. Completion time minimization for multi-UAV-enabled data collection. IEEE Transactions on Wireless Communications, 2019, 18(10): 4859–4872.
9. Nuankaew P, Temdee P. Matching of compatible different attributes for compatibility of members and groups. International Journal of Mobile Learning and Organisation, 2019, 13(1): 4–29.
10. Fadi A L T, Deebak B D. Seamless authentication: for IoT-big data technologies in smart industrial application systems. IEEE Transactions on Industrial Informatics, 2020, 17(4): 2919–2927.
11. Lu Y. Artificial intelligence: A survey on evolution, models, applications and future trends. Journal of Management Analytics, 2019, 6(1): 1–29.
12. Xu L D, Duan L. Big data for cyber physical systems in industry 4.0: a survey. Enterprise Information Systems, 2019, 13(2): 148–169.
13. Ma Z, Xiao M, Xiao Y,Pang Z, Poor H V, Vucetic B. High-reliability and low-latency wireless communication for Internet of Things: Challenges, fundamentals, and enabling technologies. IEEE Internet of Things Journal, 2019, 6(5): 7946–7970.
14. Daquino M, Heibi I, Peroni S, David S. Creating RESTful APIs over SPARQL end points using RAMOSE. Semantic Web, 2022, 13(2): 195–213.
15. Ma S P, Lin H J, Hsu M J. Semantic restful service composition using task specification. International Journal of Software Engineering and Knowledge Engineering, 2020, 30(06): 835–857.
16. Alshraiedeh F S, Katuk N. A URI parsing technique and algorithm for anti-pattern detection in RESTful Web services. International Journal of Web Information Systems, 2021, 17(1): 1–17.
17. Frozza A A, Mello R S. JS4Geo: a canonical JSON Schema for geographic data suitable to NoSQL databases. GeoInformatica, 2020, 24(4): 987–1019.
18. Povzner A, Mahajan P, Gustafson J, Rao J, Juma I, Min F, et al. Kora: A Cloud-Native Event Streaming Platform For Kafka. Proceedings of the VLDB Endowment, 2023, 16(12): 3822–3834.
19. Hassan A A, Hassan T M. Real-time big data analytics for data stream challenges: an overview. European Journal of Information Technologies and Computer Science, 2022, 2(4): 1–6.

20. Kekevi U, Aydın A A. Real-time big data processing and analytics: Concepts, technologies, and domains. Computer Science, 2022, 7(2): 111–123.
21. Baruah N, Kraft P, Kazhamiaka F, Bailis P, Zaharia M. Parallelism-optimizing data placement for faster data-parallel computations. Proceedings of the VLDB Endowment, 2022, 16(4): 760–771.
22. Kathare N, Reddy O V, Prabhu V. A comprehensive study of Elasticsearch. International Journal of Science and Research (IJSR), 2020, 4(11): 34–38.
23. Saabith A L S, Vinothraj T, Fareez M. Popular Python libraries and their application domains. International Journal of Advance Engineering and Research Development, 2020, 7(11): 19–21.
24. Buzhin I G, Derevyankin A Y, Antonova V M, Perevalov A.P., Mironov Yu B. Comparative Analysis of the REST and gRPC Used in the Monitoring System of Communication Network Virtualized Infrastructure. T-Comm-Телекоммуникации и Транспорт, 2023, 17(4): 50–55.
25. Jamzuri E R, Mandala H, Analia R, Susanto S. Cloud-based architecture for Yolov3 Object Detector using gRPC and Protobuf. Jurnal Teknik Elektro, 2022, 14(1): 18–23.
26. Karcher N, Gebauer R, Bauknecht R, Illichmann R, Sander O. Versatile configuration and control framework for real-time data acquisition systems. IEEE Transactions on Nuclear Science, 2021, 68(8): 1899–1906.
27. Corradi A, Di Modica G, Foschini L, Patera L, Solimando M. SIRDAM4. 0: A support infrastructure for reliable data acquisition and management in industry 4.0. IEEE Transactions on Emerging Topics in Computing, 2021, 10(3): 1605–1620.
28. Vyas B. Optimizing Data Ingestion and Streaming for AI Workloads: A Kafka-Centric Approach. International Journal of Multidisciplinary Innovation and Research Methodology, ISSN: 2960–2068, 2022, 1(1): 66–70.
29. Zakutynskyi I, Rabodzei I. Microservice Communication for IoT-based Systems. Architecture Review and Performance Test. Electronics and Control Systems, 2022, 4(74): 73–78.
30. Gulzar, B., Ahmad Sofi, S., &Sholla, S. (2024). Cognitive Transformation in Personal IoT: Pioneering Intelligent Automation. Cyber-Physical Systems, 11(2), 183–240.
31. Kumar A, Sivasubramaniam A, Zhu T. Split RPC: A {Control+ Data} Path Splitting RPC Stack for ML Inference Serving. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 2023, 7(2): 1–26.
32. Kaur A, Ayyagari S, Mishra M, Thukral R. A Literature Review of Device-to-Device Data Exchange Formats for IoT Applications. Journal of Intelligent Systems and Computing, 2020, 1(1): 1–10.
33. Atri P. Enabling AI Work flows: A Python Library for Seamless Data Transfer between Elasticsearch and Google Cloud Storage. J Artif Intelligent Machine Learning & Data Science, 2022, 1(1): 489–491.
34. Reddy V M, Nalla L N. Enhancing Search Functionality in E-commerce with Elasticsearch and Big Data. International Journal of Advanced Engineering Technologies and Innovations, 2022, 1(2): 37–53.
35. Vorobyeva J, Delayo D R, Bender M A, Colton M F, Pandey P, Phillip C A, et al. Using advanced data structures to enable responsive security monitoring. Cluster Computing, 2022, 25(4): 2893–2914
36. Liu C, Liu Y H, Liu J, Bierig R. Search interface design and evaluation. Foundations and Trends§in Information Retrieval, 2021, 15(3–4): 243–416
37. Novo O, Francesco M D. Semantic interoperability in the IoT: Extending the web of things architecture. ACM Transactions on Internet of Things, 2020, 1(1): 1–25.