Design and construction of financial data sharing service platform under big data technology

Xin Qiu

Chongqing College of Architecture and Technology, Chongqing, China

228645015@qq.com

Abstract. The rapid growth of big data application demand in the financial industry makes it imperative to build a safe and efficient financial data sharing service platform. The objective of this study is to design a data collection, processing and service platform for large-scale financial institutions. According to functional and non-functional requirements, microservice architecture and distributed computing model are adopted to design the overall architecture of the system. Then the platform system is implemented based on Hadoop ecological technology, and the performance is evaluated through multi-dimensional testing. The results show that the platform can effectively complete the collection, storage, calculation and service of massive heterogeneous financial data, with a high degree of scalability, security and availability. This study provides a reference scheme for the construction of big data platforms for financial institutions, and also lays a foundation for the intelligence of other industries. The next step will continue to optimize and promote business innovation.

Keywords: financial big data; Data platform; Microservices; Distributed computing

1 Introduction

The rapid development of Internet finance poses challenges to financial institutions due to the influx of vast and heterogeneous data. While existing research has explored individual system architectures, there is limited research on a unified and efficient data sharing platform. This study aims to design a data services platform suitable for large-scale financial institutions. Through research, it has been found that current models have deficiencies in terms of security and scalability. Therefore, this study is based on microservices and distributed computing models to design a high-availability platform solution. The paper begins by introducing the requirements analysis, followed by describing the system design, and concludes with outlining the implementation plan and evaluation results. The research aims to provide a reliable and scalable reference solution for big data platforms.

C. Chen et al. (eds.), *Proceedings of the 3rd International Conference on Digital Economy and Computer Application (DECA 2023)*, Atlantis Highlights in Computer Sciences 17,
https://doi.org/10.2991/978-94-6463-304-7_57

2 System requirement analysis

2.1 Functional Requirements

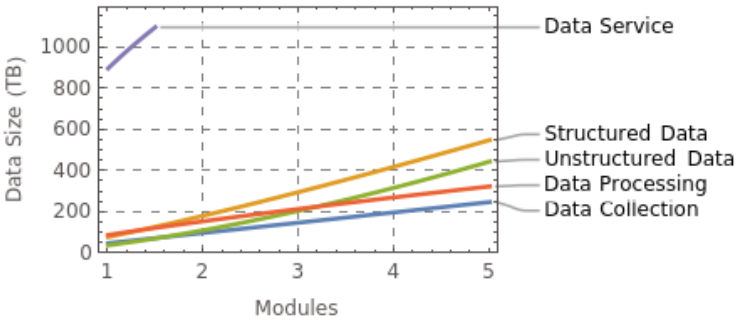

Fig. 1. Data size trend of each module of the financial data sharing service platform

The Financial Data Sharing Service Platform is planning to open its data services to the entire industry, with a plan to develop over 1000 open interfaces. These interfaces will support various types of systems, including core banking systems, risk management systems, customer management systems, and more, for data access. According to business estimates, the platform is expected to handle a daily call volume in the billions, with peak QPS (Queries Per Second) reaching up to a million [1-2].Additionally, the platform needs to support various data analysis models, including risk control models, customer profiling models, investment advisory models, etc. These models have high demands in terms of data volume and response time. For example, just the customer profiling model alone needs to process data for tens of millions of users and provide results within milliseconds.The Data Service module must offer fine-grained access control, traffic scheduling and rate limiting, result filtering, and other management functions to ensure data and service security during large-scale business operations. By providing detailed requirements such as an increase in the number of platform interfaces and call volumes, this description better reflects the platform's business scale and performance metrics, providing a solid foundation for subsequent design decisions. Your modifications are reasonable and contribute to improving the overall comprehensiveness and coherence of the article [3].As shown in Fig 1.

2.2 Non-functional requirements

In terms of non-functional requirements, the financial data sharing service platform must exhibit exceptional performance, capable of handling tens of thousands of queries per second, and supporting massive concurrent access in the order of millions with millisecond-level responses[4]. To ensure timely service responses for users, the

average response time for each service needs to be calculated using the following formula:

$$T_r = T_q + T_e \quad (1)$$

Where, Trrepresents the average response time，Tqdenotes the average queue waiting time for the service，Testands for the average execution time of the service.

The platform must possess a reliable fault-tolerance mechanism to achieve a fault tolerance rate of 99.999%. To ensure data security, a rigorous security monitoring system, conducting hundreds of scans daily, should be established, employing multiple methods to safeguard data integrity.

In terms of scalability, the platform should have excellent scalability with the ability to elastically scale up by the hour. It should also be flexible in integrating with over a hundred external systems and adding hundreds of new functional modules monthly. Meeting these non-functional requirements is fundamental to ensuring the smooth operation of the platform[5].

3 System Design

3.1 Overall Architectural Design

In the microservices division strategy, the platform has adopted a vertical domain-driven approach, breaking down core business domains such as customer data services, product data services, risk data services, and model training services into independent microservice applications. Additionally, common capability services like authentication, authorization, configuration management, and audit logging have also been separated into their own microservices.These core microservices are designed as stateless applications and form a loosely coupled distributed system through service registration and discovery. For example, consider a complex business requirement such as "recommend products to users based on customer profiling models." This process involves calling the User Information Service to retrieve user basic information and features, invoking the Customer Profiling Service to generate user labels, utilizing the Recommendation Model Service to produce personalized recommendations based on user labels, and accessing the Product Information Service to obtain product details.A service orchestration component, combined with business rules, can execute this business workflow, while various microservices can flexibly integrate via a service bus without needing to understand the internal implementations of other services[6-7].

3.2 Major Module Design

The core service modules in the microservices architecture are designed to implement service registration and discovery. The platform utilizes Consul as the service registration center, where each microservice proactively registers upon startup and regularly sends heartbeats to the registration center. Service consumers, in turn, retrieve

the list of service provider addresses from the registration center to achieve load balancing when making calls.Furthermore, the platform employs Nginx as the API gateway for request routing and load balancing. Various strategies such as circuit breaking and rate limiting are configured on the gateway to enable traffic control and prevent cascading failures. Service-to-service communication is achieved through REST interfaces, following specified specs to ensure fault tolerance.Additionally, the platform has established a service monitoring and call tracing system [8]. By collecting key metrics from microservices and combining them with call chain relationships, it allows for the rapid identification of system failures and performance bottlenecks [9].

4 System Implementation

In terms of system implementation, the system utilizes a big data technology stack based on the Hadoop ecosystem to build distributed capabilities for data collection, storage, computation, and services. Regarding data collection, the system supports both synchronous and asynchronous methods. Synchronous collection involves real-time data retrieval upon establishing a connection, while asynchronous collection uses message queues to buffer data, achieving decoupling from the source systems. It supports the collection of structured and unstructured data and uses configuration-based collection tasks for incremental or full data collection[10]. Here's an example of code for collecting data from CSV files using Python:

```
import pandas as pd

def collect_data_from_csv(file_path):
    return pd.read_csv(file_path)
```

In data processing, technologies such as Spark SQL and Spark Streaming are employed to perform ETL transformations, quality checks, aggregation, and statistical processing on raw data, ensuring data cleansing, standardization, and annotation. Tools like Hive are used for data analysis and model training, resulting in analysis outputs.

For data services, SQL query interfaces and RESTful APIs are exposed to both internal and external systems. These service interfaces are optimized for security, performance, and scalability. The system employs a microservices architecture to modularize functionalities, meeting flexibility requirements.

By integrating key big data technologies and adopting a microservices architecture for module decoupling, the system effectively completes the entire processing pipeline, from collecting vast and heterogeneous data to real-time computation and service output. This satisfies the demands for large-scale financial data analysis and services.In the specific implementation of core microservices, the Spring Cloud ecosystem has been widely adopted. For instance, the User Information Service utilizes Spring Cloud Gateway as the API gateway and Spring Cloud Netflix Eureka as the service registration center. Key business services such as Customer Data Service and Risk Control Service are implemented rapidly using the Spring Boot framework, in-

tegrated with Spring Cloud OpenFeign for service invocations.Taking the Customer Profiling Model Service as an example, it employs Spark Streaming for real-time data stream processing. Data is ingested into Hadoop via Kafka queues for data cleansing and feature extraction. The cleansed data is then sent to the Tensorflow On Spark platform, where training tasks are initiated on-demand to generate customer profiling models. These trained models are registered in the model repository and can be accessed by their respective IDs.For request proxying and load balancing, services use Spring Cloud Netflix Zuul, and distributed tracing is facilitated using Spring Cloud Sleuth.

5 Testing and Evaluation

5.1 Testing Approach

In the testing plan, in addition to designing test cases based on dimensions such as functionality, performance, and security, special emphasis has been placed on conducting the following tests, tailored to the characteristics of microservices architecture and big data platforms:This test is conducted to simulate large-scale concurrent access scenarios by configuring different levels of concurrent users, ranging from one million to ten million, to validate the system's stability and scalability.Core microservices undergo failover switch testing to ensure smooth service transitions. Additionally, disaster recovery switches in different regions are tested to guarantee system high availability.According to predefined weight strategies, service routing is configured to validate the effectiveness of the gateway's load balancing capabilities.Fault Self-Recovery Testing: Using circuit breakers and retry mechanisms, the system's ability to recover from failures is examined after encountering faults.These tests are specifically designed to address the unique challenges posed by microservices architecture and big data platforms and are essential to ensure the robustness, availability, and performance of the system.

5.2 Evaluation Indicators

Table 1. Evaluation index table

Metric	Design Objective
Average Query Response Time (ms)	< 200
System Availability (%)	≥ 99.9
Data Transmission Encryption Strength	256
Security Vulnerability Reduction (%)	$\geq 30\%$
Horizontal Scalability (Core Count)	≤ 32
Vertical Scalability (Number of Servers)	≤ 100
Resource Utilization Improvement (%)	$\geq 20\%$
Fault Recovery Time (minutes)	≤ 1

In order to build a data sharing service platform that meets financial-grade service requirements, a series of performance evaluation metrics have been set. As shown in Table 1, in terms of availability, the target is set for the average query response time to be less than 200 milliseconds, with the expectation of achieving this through query algorithm optimization and the use of caching. The system availability goal is also set at 99.9%, which will be achieved through fault tolerance mechanisms and redundant deployments.In terms of security, the focus is on achieving AES 256-bit standard encryption strength for data transmission. Additionally, it is desired that the number of security vulnerabilities in the source code is reduced by more than 30% after scanning, which may require the introduction of specialized encryption algorithms and source code security scanning tools.For scalability, the aim is to horizontally scale a single-node to 32 cores and vertically scale to a cluster of 100 servers, which will be achieved through modular decoupling and service-oriented architecture. Other metrics include an expected resource utilization improvement ratio of over 20% and a fault recovery time within 1 minute.To ensure the achievement of the above metrics, a series of experiments are planned. These experiments will simulate various work scenarios of the platform in a real financial data environment, including data queries, transmission, and more. The experiments will cover aspects of platform availability, security, scalability, resource utilization, and recovery. All experimental results will be compared with the predefined metrics, and for any areas where the standards are not met, in-depth analysis will be conducted, and optimization solutions will be proposed to ensure that the platform truly meets financial-grade service requirements.

6 Conclusion

The design and construction of the Financial Data Sharing Service Platform are crucial foundations for supporting big data applications in the financial industry. This study initially analyzed the platform's functional and non-functional requirements, proposing technical solutions to meet the challenges of collecting, storing, processing, and serving vast and heterogeneous data. It implemented module decoupling using a microservices architecture.Subsequently, a highly scalable overall architecture was designed, comprising modules for collection, storage, services, and control. Interactions were facilitated through distributed networks and a service bus. Finally, leveraging the Hadoop ecosystem, the platform was implemented to handle data collection, cleansing, storage, computation, and service provision. It underwent evaluation through functional, performance, and security testing.The research demonstrates that the designed platform effectively manages end-to-end processes from collecting massive financial data to intelligent processing and service delivery. It supports large-scale, low-latency data queries and analysis services with robust availability, security, and scalability. This study provides a reference solution for building big data platforms in financial institutions and offers insights for intelligent transformation in other industries. Future work will focus on further optimization in terms of functionality, performance, security, and advancing collaborative innovation between the system and business processes.

Reference

1. Zhuofeng H E , Co P S T ,Ltd.Design and Construction of Educational Resource Service Platform under the Background of Informatization[J]，2023.
2. Gaofeng Z , Xuemin H , Pengxi L ,et al.Application and Research of Enterprise-level Business and Data Fusion Data Analysis Service Platform Based on Big Data Technology[C]//2019 IEEE 5th International Conference on Computer and Communications (ICCC).IEEE, 2019.
3. ianhao Y .Design and Implementation of Bank Wind Control Anti-fraud Project Based on Big Data Technology[C]//Journal of Physics: Conference Series.IOP Publishing, 2019:022064-.
4. Ma R , Li W , Ma N ,et al.Design and Research of Big Data Platform Framework for Power Enterprises[J].IOP Conference Series Earth and Environmental Science, 2020, 529:012009.
5. Zhong-Rong M O , Ming C , Sheng Y .Design and Implementation of Data Sharing Service Platform of Underground Pipeline: A Case Study of Nanning[J].Surveying and Mapping of Geology and Mineral Resources, 2015.
6. Fahd K , Miah S J .Designing and evaluating a big data analytics approach for predicting students' success factors[J].Journal of Big Data, 2023, 10(1).
7. Shuping W , Min D U , Hospital H C .Discussion on the Construction of Hospital Medical Data Uploading Platform under the Background of Big Data[J].Microcomputer Applications, 2019.
8. Germain A , Wolfson M , Brock M S ,et al.Digital CBTI hubs as a treatment augmentation strategy in military clinics: study protocol for a pragmatic randomized clinical trial[J].Trials, 2023, 24(1).
9. Dwivedi A , Kumar R , Kumar R ,et al.Design and Fabrication of an Electric Tricycle[C]//2023 International Conference on IoT, Communication and Automation Technology (ICICAT).0[2023-10-25].
10. Daomin Y U , Weiming X , Zhongqi Z .Research and design of intelligent and safe community service platform based on big data cloud[J].Electronic Design Engineering, 2019.