GEORGE BROWN PRESS
JOURNALS, BOOKS & PROCEEDINGS

International Journal of Engineering Advances

Article

High Reliability Architecture and Compliance Design of Enterprise Level Financial Infrastructure

Yue Qi [1,*]

[1] School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213, USA
* Correspondence: Yue Qi, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213, USA

Abstract: Along with the rapid evolution of the modern financial industry, enterprise-level financial infrastructure must increasingly satisfy the dual demands of high-level security and stringent regulatory compliance within the financial sector. This article focuses on the systematic construction of an intelligent and highly efficient infrastructure, integrated with a comprehensive legal and regulatory compliance framework. To significantly enhance system reliability, the study utilizes AI-driven load scheduling and predictive analytics to achieve dynamic resource allocation and intelligent capacity management. Simultaneously, the integration of edge computing technology improves real-time response speeds and supports a robust fault-tolerant, self-healing operational mode, ensuring continuous service availability. In terms of compliance and risk governance, this research designs a standardized permission control model, an automated compliance auditing and response algorithm, and a trusted audit framework capable of full-chain tracking. Through the strategic application of these advanced technologies, the framework achieves enhanced data security and operational transparency, effectively building a future-oriented financial infrastructure. This research provides both a theoretical foundation and a practical implementation roadmap for financial institutions seeking to optimize their underlying technical architecture while maintaining rigorous adherence to industrial standards and safety protocols.

Keywords: financial infrastructure; high reliability architecture; compliance design; edge computing; fault self-healing

1. Introduction

In the current era of rapid financial digitalization, enterprise-level financial infrastructure faces unprecedented dual challenges. On one hand, the soaring volume of daily transactions requires systems with extreme throughput capabilities; on the other hand, increasingly stringent regulatory requirements necessitate sophisticated internal controls and risk management protocols. Under these circumstances, ensuring information system stability and strict compliance has become the cornerstone of business continuity and comprehensive information security. Traditional architectural frameworks often struggle to cope with the immense pressures of high concurrency and massive traffic fluctuations, resulting in performance bottlenecks and increased operational risks. Consequently, there is an urgent and critical need to explore and adopt innovative technological approaches to enhance system performance and overall responsiveness.

This paper primarily focuses on the design of a highly efficient and stable architectural framework based on the fundamental principles of intelligent load scheduling, decentralized edge computing, and automated fault self-recovery mechanisms. These technical components work in synergy to optimize resource utilization and ensure system resilience under stress. Simultaneously, addressing the necessity of regulatory adherence, this research proposes a robust governance layer based on rule-driven access control, automated compliance inspection algorithms, and a multi-

Received: 04 November 2025
Revised: 25 December 2025
Accepted: 09 January 2026
Published: 12 January 2026

dimensional trusted audit system. By integrating these advanced methodologies, this paper provides a comprehensive infrastructure design path that effectively balances high-performance operational goals with rigorous compliance standards. Through detailed analysis and structural optimization, the proposed framework aims to offer a reliable technical roadmap for the sustainable development and digital upgrading of modern financial information systems.

2. Overview of Enterprise Financial Infrastructure

2.1. Characteristics of Financial Infrastructure

Enterprise level financial service platforms have some unique attributes that together ensure the stability of financial services and the reliability of the system. One of their core attributes is high availability. Because financial transactions are highly time sensitive, a few minutes or even a few seconds of downtime can cause huge losses. Redundant design and automated fault recovery mechanisms need to be adopted to ensure the continuous operation of the system and avoid service interruptions. With the development and growth of business, the continuous increase in transaction and data volumes make efficiency and scalability another two essential factors. To ensure efficiency, the system must be capable of handling high transaction concurrency and fulfilling real-time data query requirements; Meanwhile the system needs to provide elastic scalability to support the growing transaction and data volumes, ensuring smooth operation in various business environments. Due to the involvement of sensitive data in the financial system, protecting data security and confidentiality has become a fundamental requirement [1]. Therefore, financial infrastructure needs to adopt strict security measures, such as data encryption, user authentication, and authorization management, to resist external attacks and prevent internal information leaks.

2.2. Core Components of Financial Infrastructure

The trading engine is a core component of financial infrastructure, responsible for processing transaction requests and matching orders. Excellent processing ability and extremely short response time are required, especially in high-frequency trading and real-time market environments [2]. Its performance directly affects trading efficiency and liquidity of funds. The core modules and functions of enterprise level financial infrastructure are shown in Table 1.

Table 1. Core modules and functions of enterprise level financial infrastructure.

Module name	Technology composition	Description of core functions
Load scheduling system	LSTM model, scheduling engine, feedback mechanism	Realize real-time prediction and resource allocation to alleviate resource bottleneck
Edge computing nodes	Edge container, CDN cache, local gateway	Reduce latency and improve high-concurrency transaction processing capabilities
Fault tolerance and self-healing mechanisms	Heartbeat detection, master/backup switching, service grid	Fault quick detection and recovery to ensure service continuity
Permission control framework	RBAC policy, authentication gateway	Assign access rights to roles to prevent overstepping and disclosure
Compliance detection system	Rule engine, behavior recognition model	Detect abnormal operation behavior and respond automatically
Audit trace system	Distributed log system, timestamp mechanism	Record the whole life cycle of transactions and support regulatory audit

For data generated on a large scale in financial business, such as transaction data, customer information, market trends, etc., data storage systems need to have the ability to process a large amount of data while ensuring data security to avoid data theft or leakage. Usually, distributed databases and big data platforms are used to achieve efficient access and storage of data. The network communication system is a component of financial infrastructure, serving as a channel for data transmission between various systems, with high concurrency and low latency transmission capabilities, while ensuring the security of the data transmission process and preventing data tampering and leakage. The security defense system ensures the security of transactions and data through encryption technology, identity authentication, and access control, and predicts potential risks in real time to protect the system from hacker intrusion and data leakage threats.

3. High Reliability Architecture Design for Enterprise Level Financial Infrastructure

3.1. Intelligent Driven Load Scheduling Design

The design of the intelligent load scheduling system revolves around a dynamic resource allocation mechanism, mainly consisting of four functional modules: real-time status monitoring, predictive analysis engine, scheduling decision center, and feedback adjustment mechanism (the intelligent load scheduling process is shown in Figure 1).

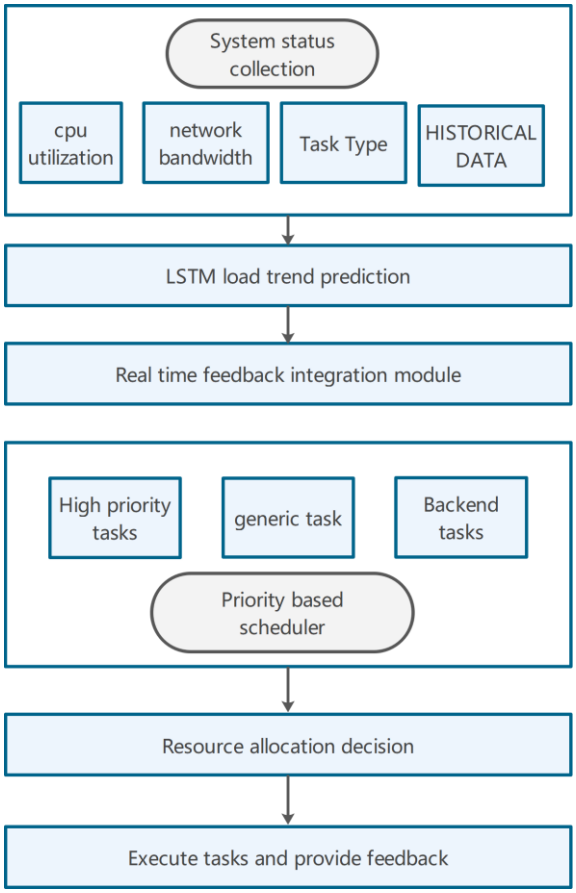

Figure 1. Intelligent Load Scheduling Flow Chart.

In the system monitoring phase, the load information of each computing node (such as CPU utilization, task queue size, network bandwidth utilization, etc.) is taken as input and converted into time series data, which is transmitted to the long short-term memory neural network prediction model in the prediction phase to generate a multidimensional time series prediction model for predicting the load in the future [3].

The predicted results will be fed into the scheduling strategy engine, which uses a dynamic weight based allocation algorithm. The formula is as follows:

$$R_t = f(L_t, C_t, N_t) \quad (1)$$

Among them, R_t is the amount of resources allocated at time t, L_t is the current load at time t, C_t is the system's computing power, and N_t is the network bandwidth. This mathematical expression constructs a flexible allocation of computing resources based on parameters such as system operating load, processor performance, and communication link quality.

In order to better utilize resources and obtain higher quality services, a task classifier has been established, which will label tasks based on their attributes and assign corresponding priorities to them. The scheduling engine will match the current resource status through a priority weight matrix in each scheduling cycle to achieve fine-grained resource allocation. The feedback system can flexibly adjust certain parameters in these feedback data, such as time window size, weight coefficients, etc., so as to quickly respond to sudden high loads and architectural changes, react quickly, and adjust response strategies.

3.2. Fast Response Structure Supported by Edge Computing

It can be seen from the whole system architecture of edge computing that the design needs to distinguish the functions of the main control end and each side control end. The main control end is responsible for the core transaction logic, the main data warehouse and cross regional data fusion tasks, and each side control end is responsible for local transaction processing and data caching [4].

Each edge site deploys local computing resources and lightweight database containers, which are managed uniformly through Kubernetes or edge native platforms. This architecture also introduces a service registry that can automatically direct user requests to the nearest edge site based on geographic location.

In order to improve efficiency and reduce the load on the core network, we have developed a multi-layer caching mechanism (L1-L3), which first attempts to find user data in the L1 cache, and then automatically adjusts the TTL according to its update frequency to ensure the timeliness of the data. The estimation algorithm for the model is given below:

$$T_{total} = T_{edge} + T_{network} \quad (2)$$

In the design, T_{total} is the total response time, T_{edge} is the time it takes to process requests at the edge node, and $T_{network}$ is the transmission time of data from the edge node to the user end. By processing most of the data at the edge nodes, the total response time is greatly reduced.

Through the design of the above modules and processes, a fast response architecture for low latency, high concurrency, and high redundancy has been constructed in financial infrastructure, providing technical support for transaction level tasks.

3.3. Fault Tolerant Architecture Mechanism for Fault Self-Healing

To enable the system to have fast recovery capabilities to adapt to high trading scenarios, a closed-loop, self driving self-healing system must be established. The overall system should have an intelligent load scheduling system to ensure state linkage and service continuity between various nodes.

At the node deployment level, a distributed redundant configuration is adopted, and a deployment model based on the principle of "primary backup hot" is used in each service unit, combined with the container platform, to achieve self-management and fast restart logic of nodes. Each node regularly reports health information and cooperates with the Prometheus system to collect real-time data indicators of operating status.

The system adopts a multi-level fault detection mechanism, monitors the status of key nodes, and uses a two-layer judgment mechanism: first, read ready operations are performed at each node; then, inspections and verification operations are coducted within

each service. If multiple nodes are unable to achieve the above two operations multiple times in a row, they will be marked as a faulty state and trigger a transition action.

The fault detection and recovery time can be described by the following formula:

$$R_{fail} = f(T_{check}, P_{error}) \quad (3)$$

Among them, R_{fail} is the time for fault recovery, T_{check} is the time interval for fault detection, and P_{error} is the probability of fault occurrence.

The traffic access layer dynamically adjusts the routing based on the detection results, forwards requests to nodes in good condition, and controls traffic transfer through traffic ratio to avoid traffic overload of other nodes during the transfer process. The database connection pool and caching middleware will synchronize the core context in advance to achieve seamless switching.

By triggering the automatic reconstruction of the replica through the controller, if the failure is caused by configuration, the latest container image and service template will be loaded to start the new node; If the malfunction is caused by configuration or update, call the version control system to perform YAML rollback.

4. Compliance Design of Enterprise Level Financial Infrastructure

4.1. Design of Rule-Based Permission Control Strategy

By using rule-based permission control strategies to achieve system operation permission management, the application of system operation permissions follows the steps of "role binding, rule matching, permission decision-making, operation audit", focusing on role matching models creation, permission rule matrix configuration, dynamic rule interpreter implementaion, and audit information integration.

In this role mapping creation, the system needs to set up some role categories, such as system administrator, business user, risk control auditor, and regular user, and assign appropriate permissions to different roles. Role configuration is based on the resource dimension, where each resource (such as account, transaction, record, or log) is bound to recognized behavior types (such as read, write, delete) to form a permission matrix.

Next, a rule-based permission judgment module will be implemented. After obtaining the access request, the module extracts the identity information and operation type from the request, and maps the request and access rules to each other for judgment.

Access control can be represented as follows:

$$P_{access} = f(R_{user}, A_{operation}, C_{resource}) \quad (4)$$

Among them, P_{access} is the result of access permissions, R_{user} is the user's role, $A_{operation}$ is the type of operation (such as read, write, delete), and $C_{resource}$ is the security level of resources (such as accounts, funds, transaction records). Based on the matching results of roles, operations, and resources, the system decides whether to grant permissions.

4.2. Automated Compliance Testing and Response Process Design

The automated compliance testing and response process should be designed around four stages: event collection, rule comparison, risk assessment, and action execution.

In the event collection layer, a comprehensive information collection method is introduced to monitor important resources in real-time, such as customer access history, transaction data flow, access paths, system operation instructions, etc. It is recommended to use a stream processing engine to connect to the data bus and convert various operations into standardized message streams, which are transmitted to the detection module in a timely manner.

The compliance detection module is implemented using a rule engine architecture. The configuration rule set should be able to meet various types of compliance template settings, including risk management measures in multiple fields such as KYC (customer identity verification), AML (anti money laundering), and have the ability to be easily

upgraded and replaced. The system evaluates rule matching based on each event flow, and performs corresponding logical calculations based on existing contextual parameter information to determine whether there is a possibility of violation.

The core judgment process can be expressed by the following function:

$$R_{response} = f(E_{event}, P_{policy}, T_{time}) \quad (5)$$

Among them, $R_{response}$ is the operation responded by the system, E_{event} is the event type (such as abnormal transaction, unauthorized access), P_{policy} is the relevant compliance policy (such as KYC, AML policy), and T_{time} is the response time.

At the response execution level, the system automatically triggers relevant actions based on the policy calculation results. Actions need to be configured in advance according to the event level, such as generating audit logs for low-level violations and automatically freezing accounts and reporting to the risk control interface system for high-level events such as abnormal fund flows.

4.3. Design of a Trusted Compliance Framework Supporting Audit Traceability

The design of a trusted audit traceability framework is a layered design from four aspects: operation collection mechanism, trusted generation of logs, encrypted storage of logs, and verification interface structure. The purpose is to collect all data during the operation of the financial system and record the operation process to ensure its traceability.

At the operational collection mechanism layer, the system needs to introduce log probes in each service module to automatically record important events such as user identity changes, transaction approvals, and permission changes.

On the trusted generation layer of logs, the system needs to generate independent hash signatures for each log in the form of a hash chain. When each log forms a log block, calculate the hash value of the log block and concatenate it with the hash of the previous block to form an immutable chain structure. The verification function is as follows:

$$A_{\log} = H(D_i) \oplus H(D_{i-1}) \quad (6)$$

Among them, $A_{\log}$ represents the validation digest of the log blockchain, $H(D_i)$ is the hash value of the current log data, and $H(D_{i-1})$ is the hash value of the previous log block. This mechanism ensures that any field modification will cause a full chain exception.

At the level of encrypted storage system, it is necessary to divide the diary into multiple copies for decentralized storage, and use symmetric encryption to ensure the confidentiality of the diary content, and use asymmetric keys to ensure the authenticity of the access auditor's identity. In terms of verifying the interface structure, the design provides audit retrieval based on time windows, role behavior, and service paths, and embeds log comparison and integrity verification tools.

5. Conclusion

The contemporary financial industry has established increasingly stringent standards for the high reliability and regulatory compliance of information systems, elevating the development of high-performance, highly available, and secure financial infrastructure to a core strategic priority. This study demonstrates that by integrating intelligent resource allocation, decentralized edge computing technology, and automated fault self-healing mechanisms, it is possible to ensure that infrastructure maintains stable operations and millisecond-level response times, even under extreme high-concurrency pressure. Furthermore, to address complex compliance requirements, this research has constructed a robust framework consisting of rule-based permission management and automated compliance review and disposal mechanisms. The deployment of a comprehensive compliance framework with full-chain tracking capabilities effectively safeguards data integrity and ensures strict adherence to operational standards. Looking forward, financial infrastructure must undergo continuous iteration and structural upgrading to further enhance its flexibility, reliability, and governance capabilities. Such

advancements will provide a solid technical guarantee for the intelligent transformation and digital development of the financial sector, fostering a more resilient and efficient financial ecosystem. Through the systematic optimization of both hardware resource management and software compliance logic, enterprises can better navigate the technical demands of the digital era while maintaining rigorous safety protocols.

References

1. D. Leontiev, S. Chykalova, V. Asmolov, N. Volovyk, V. Petrus, and O. Gryzodub, "Analyst qualification for compliance with normal analytical practice for pipette use," *ScienceRise: Pharmaceutical Science*, vol. 6, no. 52, pp. 68-79, 2024.
2. I. T. Liu, and A. D. Dixon, "What does the state do in China's state-led infrastructure financialisation?," *Journal of Economic Geography*, vol. 22, no. 5, pp. 963-988, 2022. doi: 10.1093/jeg/lbac009
3. K. M. Sutcliffe, "Building cultures of high reliability: Lessons from the high reliability organization paradigm," *Anesthesiology clinics*, vol. 41, no. 4, pp. 707-717, 2023.
4. N. A. Zaguir, G. H. de Magalhães, and M. de Mesquita Spinola, "Challenges and enablers for GDPR compliance: systematic literature review and future research directions," *IEEE Access*, vol. 12, pp. 81608-81630, 2024. doi: 10.1109/access.2024.3406724