Scholar Publishing Group
International Journal of Neural Network
https://doi.org/10.38007/NN.2025.040103
ISSN 2790-2692 Vol. 4, Issue 1: 23-31

Exploration of State Machine-Driven Access Control Mechanisms in Distributed Systems

Yizhou Meng

DRIVE VRI COGS 1010, Microsoft, Redmond, WA, 98052, USA

Keywords: Distributed DRC; Role division; Raft protocol optimization; Data synchronization mechanism; High availability

Abstract: Technologies such as the Internet of Things are driving exponential growth in data volume, while traditional data management systems are difficult to solve the challenges of "data chimney" and cross domain sharing due to inconsistent data standards. Data oriented architecture (DOA) focuses on data as its core and builds a data ecosystem through a data registry center (DRC). However, distributed DRC faces high availability and efficiency bottlenecks in high concurrency scenarios - traditional Raft protocol dual center solutions face issues such as storage and write congestion, and "leaderless" fault selection. High availability cluster solutions lack fault tolerance and recovery capabilities. This study proposes a distributed DRC high availability scheme based on role partitioning, which includes: dividing NameNode nodes into four roles: Leader, Inheritor, etc., clarifying read-write division of labor and fault takeover mechanism; Optimize Raft protocol to avoid "leaderless state" and shorten election time; Introducing vector clock to achieve data version conflict detection and improve synchronization efficiency. The experiment shows that the performance of the improved scheme is significantly improved: the write response delay is stable at 0.16-0.75ms (the highest in the original scheme was 4.74ms), the fault recovery time is reduced to 500-800ms (the original 1400-1700ms), and the synchronization time of the three/five machine mode is 428.4ms and 588.2ms, respectively. This solution solves the problem of multi machine data consistency through algorithm innovation, providing high-performance technical support for data sharing under DOA architecture.

1 Introduction

In the field of distributed systems, the popularization of emerging technologies such as the Internet of Things, artificial intelligence, big data, and cloud computing has driven the exponential growth of data volume. Traditional business oriented data management systems are unable to solve the problems of "data chimney" [1] and "data island" due to the lack of unified data standards and deep binding between data and business, and cannot achieve data sharing across levels, regions, systems, departments, and businesses. To this end, Data Oriented Architecture (DOA) proposes the design concept of "data as the core", which constructs a data ecosystem through Data Registry Center (DRC), Data Rights Center (DAC), and Data Security Center (DEC), with DRC as a key component responsible for data unified registration and service functions. However, although distributed DRC reduces the risk of single node failure through load balancing, it still faces challenges in high availability and efficiency in high concurrency scenarios. For example, in the dual center server solution based on the Raft protocol by Maxinfan, the Leader node needs to wait for ACK messages from all Followers before executing write requests. When high concurrency writes occur, accumulated response latency can easily lead to request congestion; The "no leader state" during the election period after a leader failure makes the system unable to respond to write requests, and traditional high availability cluster solutions have limitations in fault tolerance, recovery time, resource utilization, and other aspects. This study focuses on the exploration of "state machine driven access control mechanism in distributed systems", and optimizes the access control and consistency management of distributed DRC through state machine thinking. A high availability solution for distributed DRC based on role partitioning is proposed: clarifying node roles (such as refined division of labor for Leader and Follower) and permissions, combined with optimized Raft consistency protocol to reduce fault recovery time and achieve client seamless fault switching; Adjust the read-write permission strategy, adopt one-way data synchronization to avoid resource waste, and design data version conflict detection algorithms and distributed synchronization methods to improve read-write efficiency while ensuring data consistency. The research significance lies in two aspects: firstly, to equip DRC with more comprehensive disaster recovery solutions through role partitioning models and protocol optimization, ensuring the continuous service capability of the system; The second is to solve the problem of data consistency in multi machine states through algorithm innovation, and improve the performance and availability of distributed DRC. The paper focuses on building a high-performance and highly available distributed DRC architecture, proposing a distributed DRC model based on role partitioning, and clarifying the logic of node collaborative work; Optimize Raft protocol to achieve fast fault handling; Design data version conflict detection and synchronization methods to ensure data consistency in high concurrency scenarios; Verify the improvement of the scheme in indicators such as read and write efficiency and fault recovery time through experiments.

2 Correlation theory

2.1 Theory of Data Oriented Architecture (DOA)

Data oriented architecture (DOA) [2] is a software architecture concept centered around data, aimed at building a data ecosystem through data registration centers (DRC), data permission centers (DAC), data anomaly control centers (DEC), and data application units (DAUs). It solves the problems of "data chimneys" and "data islands" caused by inconsistent data standards and deep bundling of data and business in traditional business-oriented systems, and achieves data sharing across levels, regions, systems, departments, and businesses. The components of DOA include: DRC is responsible for unified data registration and management, defining data registration standards (covering data features, ownership, location, attributes, etc.), using structured and unstructured data automatic registration methods, combining data recognition and classification, index retrieval, and distributed deployment to build a logical data resource pool, serving as a bridge between real data and upper level applications; DAC is responsible for data access permission management, ensuring data security through user level authentication, data encryption, and authorization mechanisms; DEC is responsible for data consistency detection and exception handling in distributed environments, including conflict detection, synchronization processing, and redundancy management; DAUs provide services to upper level applications through application programming interfaces, supporting secondary transformation of application units to quickly meet diverse needs. DRC, as a core component, follows the concept of "all data is registered in DRC, and all applications go through DRC". Through unified data registration standards, it realizes the interconnection of DRC in different fields and industries, forms a grand data resource pool, supports cross system data interconnection and sharing, and improves read and write efficiency and disaster recovery capabilities through distributed deployment.

2.2 Distributed System Theory and High Availability Architecture

Distributed systems solve the performance bottleneck of single node storage by running applications and data in collaboration with multiple servers, improving scalability, availability, and reliability. However, they face network unreliability, partitioning issues, and consistency challenges. The CAP theory states that distributed systems cannot simultaneously meet strong consistency, high availability, and partition fault tolerance, and a balance needs to be struck among the three; The BASE theory proposes achieving high availability and partition fault tolerance through final consistency, which is suitable for large-scale systems. Hadoop, as a typical distributed architecture, includes HDFS (Distributed File System)[3] and MapReduce [4] (Big Data Processing Engine) at its core. HDFS manages metadata through NameNodes and stores data blocks through DataNodes, supporting data redundancy and fast response; MapReduce achieves efficient computation through task decomposition and parallel processing. High availability clusters ensure service continuity through a failover mechanism. Common models include Active Standby, Read Write Separation, Dual Master, N+1/N+M (for multi active node deployment), N-to-1/N-to-N (for decentralized task transfer), etc. The measurement indicators include Mean Time to Failure (MTTF), Mean Time to Repair (MTTR), Mean Time Between Failures (MTBF), and System Availability. International standards classify availability levels based on annual downtime These theories and technologies together form the design foundation for highly available and consistent distributed systems.

3 Research method

3.1Analysis of High Availability Requirements and Challenges for Distributed Data Registry

Distributed Data Registry (DRC)[5]implements distributed storage based on Hadoop, with NameNode serving as the central server to process client requests. Data registration information is segmented and stored in DataNode nodes, and a multi replica mechanism is adopted to avoid loss. However, a single point of failure of the NameNode (such as hardware failure or maintenance) can lead to the overall paralysis of DRC, and its availability is highly dependent on the stability of the NameNode. To address this issue, Hadoop proposes a high availability (HA) solution and introduces an Active Standby dual NameNode architecture: the Active node provides external services, while the Standby node synchronizes status to achieve fast failover. However, there is still a single point of failure risk before the original NameNode is restored. Furthermore, the dual center server solution eliminates single point of failure through the election of three NameNodes (Leader+two Followers) and Raft consistency protocol, but introduces a new problem: when writing with high concurrency, the Leader needs to wait for ACK messages from all Followers, resulting in accumulated response latency and request congestion; During the election period after the Leader failure, DRC fell into a state of no Leader and was unable to respond to write requests. Therefore, achieving high availability in DRC requires optimization from two aspects: reducing the switching time between primary and backup, and the waiting time for multiple machines to select primary. At the same time, it is necessary to clarify node roles and permissions to reduce request processing latency and improve execution efficiency.

3.2 Four role partitioned distributed DRC high availability and high-performance model

The distributed DRC model adopts a cluster architecture based on role partitioning, consisting of NameNode nodes[6] (including four roles: Leader, Inheritor, Follower, Candidate), DataNode nodes, and clients. The NameNode queue contains one Leader, one Inheritor, and several dynamically adjustable numbers of Follower nodes. Among them, the Leader is responsible for handling client write requests, synchronizing logs to the Inheritor and obtaining ACK feedback to ensure data consistency before executing the write. Then, the Inheritor asynchronously replicates the data to the Follower, which not only ensures data consistency between the primary and backup nodes, but also reduces the Leader's waiting time to improve write efficiency; The read request is jointly executed by the Inheritor and Active Follower. Followers are divided into two states based on data version consistency: Active (consistent with the Inheritor data, can share the read) and Standby (unsynchronized data, needs to be updated and converted to Active). In terms of role transition mechanism: When a Leader fails, the Inheritor is upgraded to a Leader and the change is broadcasted. If there is a higher-level Leader after the original Leader is restored, it is converted to a Follower to avoid brain cracking; When the Inheritor malfunctions, Follower switches to Candidate to initiate the election, and switches roles based on the results after the election is over. The data model adopts a dual mode of structured and unstructured data: structured data is stored in a database table field hierarchy, while unstructured data contains metadata such as ownership and security levels; The storage model is based on a distributed file system master-slave structure, where the Leader coordinates the allocation of data packets to DataNodes and synchronizes them with other NameNode nodes through logs. The internal structure, read-write process, role transformation, and storage model of the model correspond to Figure 1, respectively

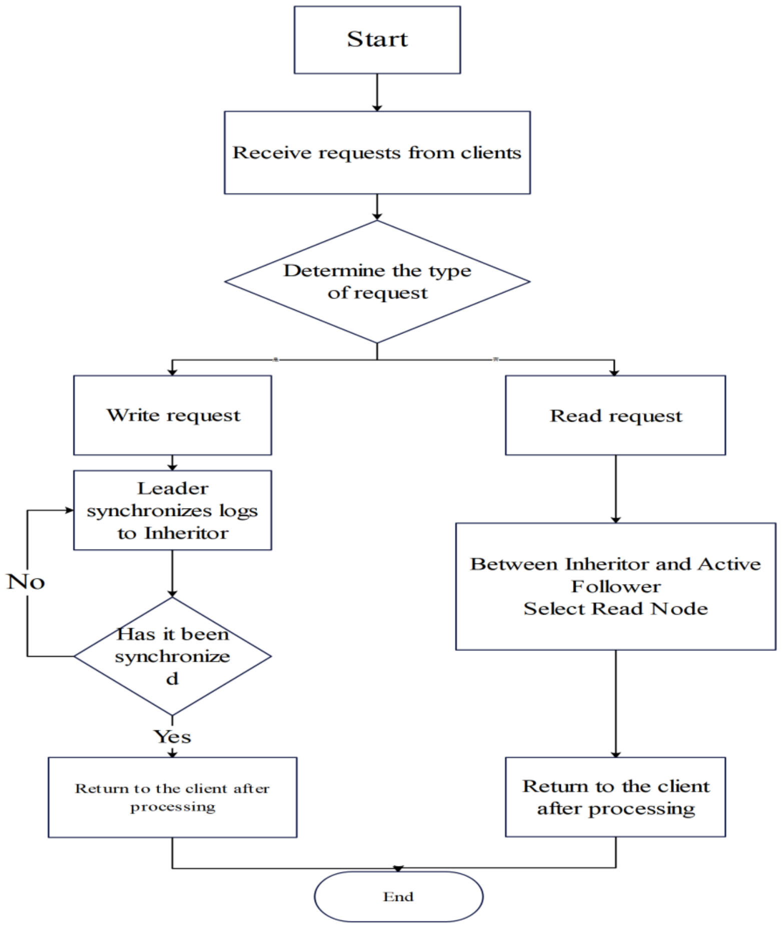
Start
Receive requests from clients
Determine the type of request
Write request
Read request
Leader synchronizes logs to Inheritor
No
Has it been synchronize d
Yes
Between Inheritor and Active Follower Select Read Node
Return to the client after processing
Return to the client after processing
End

Figure 1Flow Chart of Distributed DRC Client Request Processing Based on Role Partition

This paragraph analyzes the current situation and requirements of distributed DRC, proposes the model, and elaborates on the role permission division and data model design scheme to achieve the goals of high availability and high-performance data management.

3.3 Heterogeneous Graph Attention Financial Feature Extraction and Experimental Verification

Distributed DRC achieves high availability through fault detection and node election mechanisms, ensuring seamless takeover of services by backup nodes in the event of a primary node failure without client awareness. Fault detection adopts two modes: dual machine (Leader Inheritor) and multi machine (Inheritor Follower). In dual machine mode, the Leader synchronizes its status through active heartbeat packets within cycle T. The Inheritor initiates polling and determines that the Leader has failed when it has not received a heartbeat for a total of N times, triggering fault transfer (corresponding to Figure 2)

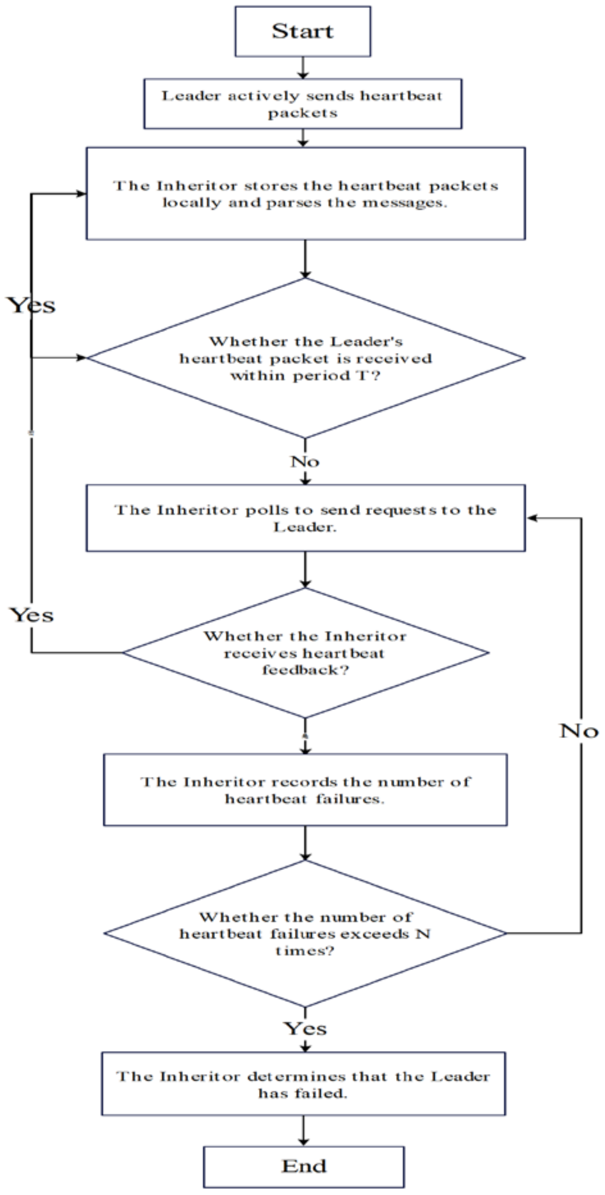
Start
Leader actively sends heartbeat packets
The Inheritor stores the heartbeat packets locally and parses the messages.
Yes
Whether the Leader's heartbeat packet is received within period T?
No
The Inheritor polls to send requests to the Leader.
Yes
Whether the Inheritor receives heartbeat feedback?
No
The Inheritor records the number of heartbeat failures.
Whether the number of heartbeat failures exceeds N times?
Yes
The Inheritor determines that the Leader has failed.
End

Figure 2 Flow chart of dual node heartbeat detection and fault determination

In multi machine mode, the Inheritor broadcasts the heartbeat to the Follower, and when more than half of the Followers have accumulated N missed heartbeats, the Inheritor election is initiated. Node election is based on the Raft algorithm optimization scheme, which introduces random clock to avoid tie votes, vector clock to compare data versions, term division mechanism to ensure temporal consistency, and defines the election result status (successful/rejected/failed/timeout). The election rules include: only Active Followers can run, Standby Followers need to sort by vector clock and select three candidates; Each candidate can cast their own vote and submit a voting request to Followers, with only one vote per node during the same term; Nodes only respond to requests with a Term value not less than their own, and candidates who fall behind need to initiate new elections. The election process consists of four steps: after more than half of the followers determine that there is no Inheritor status, the election is initiated. Based on the presence of Active Followers, a random clock is added or Standby Followers are selected. Candidates broadcast campaign information and count votes, and finally determine a new Inheritor or enter the next term based on the vote count. This mechanism supports read-write services in the absence of an Inheritor, and ensures data consistency and service continuity of the system in fault scenarios through Leader Inheritor Follower role partitioning and dynamic election.

4 Results and discussion

4.1 Distributed DRC Fault Transfer Mechanism and Key Class Design

Distributed DRC is based on the Hadoop cluster architecture, implementing inter node communication through RPC/TCP/IP protocol, and using virtual IP and VRRP protocol to achieve address drift - when the primary NameNode fails, the backup node switches the MAC address of the virtual IP to itself through ARP spoofing, achieving seamless service takeover. The fault handling mechanism consists of three major modules: fault detection, node selection, and fault transfer. Fault detection relies on the HeartbeatMessage class (including SourceIP, Destination IP, and other attributes) and the HeartbeatHandler class (including LastHeartbeatTime and Failed Number attributes), supporting dual/multi machine heartbeat monitoring; Node election implements the election process of Candidate initiating voting and Follower responding to requests through the NodeElection class (including ElectTerm attribute) and VoteResult class (including BallotTicket voting object); The failover is driven by the Failed over Handler class, which triggers the virtual IP drift program by calling the Failed over method[7], and combines with the RoleManager class (implementing the RoleState interface) to complete role permission partitioning and state transition. This mechanism ensures high availability and service continuity of distributed DRC in node failure scenarios through dynamic binding of virtual IP, heartbeat failure determination, election process control, and role status management.

4.2 Model experiment

To verify the performance of the distributed DRC high availability solution based on role partitioning, a client Leader、Inheritor、 A testing environment with three Followers and three DataNodes. The hardware adopts a third-generation Intel Xeon scalable processor (with a main frequency of 2.7GHz, a full core turbo frequency of 3.5GHz, 12G memory, and 40G dedicated hard disk) for the server and an i7-5500U CPU (8GB memory) for the client. The software is based on CentOS 6.0 system and integrates Hadoop, JDK 1.8.0, and Filezilla transmission tools. The test data covers 15000 structured MySQL data and 3000 unstructured files (such as DOCX, ZIP, XLSX, JPG, etc.), which are extracted and written into the system through a data registration engine. Three major tests for experimental design: write response time test. The client sends 1000 write requests per second for 10 minutes, records a single delay, and repeats it 20 times (with a 5-minute interval); Fault recovery time test simulates Leader/Inheritor node downtime, records the time it takes for the system to recover read-write services from the fault, and repeats 100 times (with a 5-minute interval); The data synchronization time test uses multi-threaded writing to kill the Leader node, monitor the switching time and overall synchronization time of Followers from Standby to Active state, and repeat 100 times (with a 5-minute interval). Through three sets of comparative experiments, the focus is on verifying the optimization effect of the improved scheme in terms of read-write performance, fault recovery efficiency, and data consistency achievement, providing empirical support for the availability and effectiveness of the scheme.

4.3 Effect analysis

This article conducts performance comparison tests between the role based distributed DRC model and the original distributed DRC scheme, with a focus on verifying three core indicators: write response time, fault recovery time, and data synchronization time. The testing environment is built on eight servers, and the test data is sourced from the official website of Chengdu University of Technology. The experimental design includes three sets of testing methods, and each set of experiments is repeated multiple times to take the average value to ensure reliability.

The write response time test adopts four comparative schemes: the original scheme has three nodes, the original scheme has five nodes, the improved scheme has three nodes, and the improved scheme has five nodes. Each round of the experiment lasts for 10 minutes, with write requests initiated at a frequency of 1000 times per second. The total number of requests reaches 60000, and the average of 20 tests is taken. As shown in Table 1, the original five node scheme maintained a latency of 0.2-0.5ms within 1-2 minutes, but with the superposition of requests, the Leader had to wait for confirmation from all nodes, resulting in a sharp increase in latency - the delay of the three-node scheme reached 2.946ms in the 10th minute, and the five-node scheme soared to 4.74ms. The improved scheme, however, had a stable latency of 0.2-0.4ms for the three and five nodes within 1-5 minutes, with a maximum latency of only 0.758ms. The increase in the number of nodes did not affect performance.

The fault recovery time test focuses on the three-node mode, simulating the collapse scenarios of Leader, Inheritor, and Follower nodes. The experiment was repeated 100 times, with each test lasting 5 minutes.

Table 1 Comparison of Write Response Time Test Results

Minute	Original Scheme (3 Nodes)	Original Scheme (5 Nodes)	Improved Scheme (3 Nodes)	Improved Scheme (5 Nodes)
1	0.21ms	0.224ms	0.161ms	0.196ms
2	0.38ms	0.39ms	0.19ms	0.244ms
3	0.398ms	0.482ms	0.23ms	0.27ms
4	0.432ms	0.87ms	0.218ms	0.66ms
5	0.59ms	1.48ms	0.25ms	0.39ms
6	1.1ms	2.97ms	0.22ms	0.29ms
7	1.48ms	3.56ms	0.52ms	0.33ms
8	1.729ms	3.992ms	0.758ms	0.319ms
9	2.163ms	4.12ms	0.34ms	0.42ms
10	2.946ms	4.74ms	0.32ms	0.278ms

The original plan focused on the recovery time of Leader faults between 1400-1700ms, with an average of 1639.2ms; the improved plan shortened the recovery time to 500-800ms, with an average of 724.4ms, which was 914.8ms less than the original plan due to the direct takeover by the

Inheritor. For Inheritor faults, the original plan did not affect read and write operations due to redundant Follower nodes; In the improvement plan, if there is an Active Follower, it can temporarily execute read requests, with a fault recovery time of 200-300ms and occasional peak of 739ms; if there is no Active Follower, the Leader can temporarily read requests, although the efficiency is reduced, the system can still maintain service. The data synchronization time test verifies the FullFChangeTime in the three machine and five machine modes. The experiment is repeated 100 times, each time for 1 minute. The synchronization time of the three-machine mode is 200-500ms, with an average of 428.4ms and an extreme value of 2478ms; the five-machine mode needs to synchronize more Follower nodes, and the time is extended to 300-700ms, with an average of 588.2ms and an extreme value of 4234ms. The conclusion shows that although increasing Follower nodes improves reading efficiency, the synchronization time and packet loss probability increase synchronously, and the number of nodes needs to be dynamically adjusted according to the number of read and write requests. In summary, the distributed DRC scheme based on role partitioning significantly outperforms the original scheme in terms of write response, fault recovery, and data synchronization performance, effectively solving the problems of write congestion and long recovery time, and demonstrating advantages in high availability and performance.

5 Conclusion

With the rapid development of the Internet industry, the explosive growth of multi-source heterogeneous data [8]has brought significant challenges to data sharing, mining and application. Traditional industry data management systems have limitations in cross domain data sharing scenarios[9], while DOA (Data Registry Architecture) achieves the sharing and circulation of underlying data by designing unified data registration standards. As the core module of DOA, distributed DRC (Data Registry) constructs a logical data resource pool, linking data registry centers of different scales and industries to form a larger scale DRC. With the expansion of DRC scale and the surge in data volume, high availability in high concurrency scenarios has become a key requirement, and the design of reasonable disaster recovery solutions has therefore become a research focus. This article proposes a distributed DRC high availability solution based on role partitioning, which includes three core designs: a distributed DRC model based on role partitioning (dividing NameNode nodes into four roles: Leader, Inheritor, Follower, and Candidate. Leader handles write requests, other roles handle read requests, and Inheritor takes over services in case of Leader failure to improve system availability); Distributed DRC fault handling mechanism (implementing node status monitoring through heartbeat mechanism, using optimized Raft consistency protocol to solve multi node primary selection problem, and combining virtual IP drift technology to complete primary and backup service switching, achieving client insensitive fault handling); Distributed DRC data synchronization mechanism (introducing vector clock to achieve data version conflict detection, proposing an improved data synchronization method - Leader and Inheritor can respond to clients after synchronization, without waiting for ACK messages from half of the Followers, ensuring data consistency and improving synchronization efficiency). The research focuses on four aspects: analyzing existing distributed DRC problems and proposing a role partitioning model; Research on fault handling mechanisms; Research on data synchronization mechanism; Compare the reliability and maintainability of the original plan with the high availability plan to verify feasibility. The innovation lies in the clear division of labor and collaboration in the role division model, which improves read and write performance and fault handling efficiency; The improved Raft algorithm avoids the absence of a Leader state, and the Inheritor switches directly to reduce election time; The conflict detection and synchronization method based on vector clock[10]ensures data consistency.

References

[1] *Chen W, Zuo J, Su S, et al. A Privacy-Preserving Across-Institution Archives Utilization Framework and Implementation[J]. 2024 IEEE Cyber Science and Technology Congress (CyberSciTech), 2024:138-145. DOI:10. 1109/cyberscitech64112. 2024. 00031.*

[2] *Erics Z, Cirulis A. A Proposal foranAdaptive Compiler Architecture andDesign forPrecision andScale Aware Compilation Using Data-Oriented Design[C]//International Conference on Human-Computer Interaction. Springer, Cham, 2025. DOI:10. 1007/978-3-031-93848-1_13.*

[3] *Information V F A, Hnumanthappa L P, Information V F A, et al. Fractional social optimization-based migration and replica management algorithm for load balancing in distributed file system for cloud computing[J]. 2024.*

[4] *Chen X. Research on AI-Based Multilingual Natural Language Processing Technology and Intelligent Voice Interaction System[J]. European Journal of AI, Computing & Informatics, 2025, 1(3): 47-53.*

[5] *Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters[C]//Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6. USENIX Association, 2004. DOI:10. 1145/1327452. 1327492.*

[6] *Banerji N, Choudhury B, Choudhury S. Design of a Context-Aware Distributed Service Registry and Discovery Mechanism for IoT[J]. IEEE Internet of Things Journal, 2024, 11(15):12. DOI:10. 1109/JIOT. 2024. 3395155.*

[7] *Wu X, Bao W. Research on the Design of a Blockchain Logistics Information Platform Based on Reputation Proof Consensus Algorithm[J]. Procedia Computer Science, 2025, 262: 973-981.*

[8] *Li W. Building a Credit Risk Data Management and Analysis System for Financial Markets Based on Blockchain Data Storage and Encryption Technology[C]//2025 3rd International Conference on Data Science and Network Security (ICDSNS). IEEE, 2025: 1-7.*

[9] *Zhang K. Research on the Application of Homomorphic Encryption-Based Machine Learning Privacy Protection Technology in Precision Marketing[C]//2025 3rd International Conference on Data Science and Network Security (ICDSNS). IEEE, 2025: 1-6.*

[10] *Tang X, Wu X, Bao W. Intelligent Prediction-Inventory-Scheduling Closed-Loop Nearshore Supply Chain Decision System[J]. Advances in Management and Intelligent Technologies, 2025, 1(4).*

[11] *Huang, J. (2025). Research on Resource Prediction and Load Balancing Strategies Based on Big Data in Cloud Computing Platform. Artificial Intelligence and Digital Technology, 2(1), 49-55.*

[12] *Zhou Y. Cost Control and Stability Improvement in Enterprise Level Infrastructure Optimization[J]. European Journal of Business, Economics & Management, 2025, 1(4): 70-76.*

[13] *Chen X. Research on the Integration of Voice Control Technology and Natural Language Processing in Smart Home Systems[J]. European Journal of Engineering and Technologies, 2025, 1(2): 41-48.*

[14] *Chen M. Research on the Application of Privacy-enhancing Technologies in AI-driven Automated Risk Detection Systems[J]. Advances in Computer and Communication, 2025, 6(4).*

[15] *Lyu N. Adaptive Generative AI Interfaces via EEG-based Cognitive State Recognition[J]. Advances in Computer and Communication, 2025, 6(4).*