Scholar Publishing Group
International Journal of Multimedia Computing
https://doi.org/10.38007/IJMC.2026.070106
ISSN 2789-7168 Vol. 7, Issue 1:42-51

Construction of a Cloud-Native High-Performance Service Engineering System for Real-Time Decision-Making Platforms

Jiahe Chen

Computer Science, Columbia University in the City of New York, New York, 10027, USA

Keywords: Cloud-native database, decoupled memory architecture, persistent memory optimization, cross-regional distributed transactions, and elastic resource scaling

Abstract: In the cloud-native environment, microservice architecture faces dual challenges: ambiguity in service configuration semantics and the reliance on manual experience for change generation. Traditional traffic limiting and resource scheduling methods suffer from static nature, insufficient global optimization, and lack of collaboration. Research Method: A microservice orchestration framework based on LLM is proposed. LLM is utilized to parse service configuration semantic information, achieving precise mapping from natural language to configuration parameters. Reinforcement learning (such as DQN model) is integrated to dynamically generate traffic limiting strategies and container resource adjustment schemes, supporting both non-interactive and interactive collaborative adjustment modes. Research Results: Under Workload1 and Workload2 loads, the request success rate is improved by an average of 57% compared to static methods and by 32%-56% compared to the SPCB algorithm. In scenarios with dynamic resource changes, the response time violation rate is further reduced through state space expansion (incorporating total resource amount and arrival rate), maximizing the request success rate. Conclusion: This framework leverages LLM semantic parsing and reinforcement learning strategies to achieve intelligent collaboration between microservice request connection limits and container resources, enhancing system stability and resource utilization. It provides a new path for cloud-native microservice orchestration.

1 Introduction

Driven by the dual forces of digital transformation and exponential growth in data volume, database systems[1]have become the core backbone of modern information systems [2], supporting the underlying logic of key operations, decision-making processes, and customer interactions in enterprises. The global market size of database management systems is expected to reach $125.6 billion by 2026, with a compound annual growth rate of 12.4%. Their importance is particularly prominent in applications such as ERP, CRM, and e-commerce, where 91.9% of enterprises have achieved measurable value through data and analytics investments. With the maturity of cloud computing technology, cloud-native databases have emerged, exhibiting significant advantages in

cross-regional distributed architecture, pay-as-you-go models, and dynamic load adaptation, thanks to their characteristics such as containerization, resource elasticity, and high reliability. Traditional databases face challenges in resource expansion, performance optimization, and cross-data center consistency: static resource allocation leads to low utilization, cross-WAN communication delays constrain real-time response, and consensus protocols[3] (such as Paxos) suffer from master node bottlenecks and load imbalance issues in multi-data center scenarios. This study aims to build a highly available cloud-native database system that supports independent and elastic resource expansion, addressing pain points such as the inability to independently expand computing/memory, cross-domain performance optimization, and adaptation to new hardware (such as persistent memory). By designing remote data pathways that are aware of database characteristics and adapting the ARIES error recovery protocol to a split architecture, dynamic expansion of computing and memory is achieved. Optimizing the storage hierarchy in conjunction with persistent memory characteristics enhances cost-effectiveness and persistent path efficiency. Developing a cross-consensus group leadership migration mechanism and transaction protocol integration scheme reduces cross-regional communication delays and ensures global consistency. The contributions of this study include proposing a universal high-speed resource independent recovery split-memory architecture, software and hardware co-design based on persistent memory, and a cross-domain optimization scheme supporting row-level multi-write and efficient transaction execution, ultimately building a high-performance, high-availability database service prototype system for global users.

2 Correlation theory

2.1 Cloud-native database dual-layer architecture

In the data-driven digital era, enterprise demands are propelling database systems towards deep adaptation to cloud environments. Cloud-native databases, as systems specifically designed for cloud computing [4], fully leverage the characteristics of cloud platforms to achieve seamless cloud migration while retaining existing application programming interfaces (APIs). Developers can leverage the advantages of cloud platforms without modifying code. Their core advantage lies in high scalability - relying on the virtually unlimited computing and storage resources of cloud platforms, resources can be dynamically adjusted to achieve high performance and high availability.From an architectural perspective, cloud-native databases adopt a dual-layer design consisting of macro and micro layers. At the macro level, the system is deployed across multiple data centers, with each data center possessing independent hardware and software resources. The distributed architecture enhances system performance and fault tolerance, ensuring that other nodes can still operate normally in the event of a single data center failure. Cross-data center interconnection is achieved through advanced consensus protocols (such as Paxos) in a wide area network, guaranteeing data consistency and availability. At the micro level, within a single data center, resource pooling solutions break traditional boundaries of standalone resource allocation, distributing CPU, memory, storage, and other resources across different resource pools and interconnecting them through high-speed networks. This significantly improves resource utilization and supports flexible and independent resource expansion. This architecture enables cloud-native databases to provide users with a consistent service experience globally, allowing users to enjoy fast response and reliable services regardless of their location, while adapting to the real-time needs of geographically dispersed user groups.

2.2 Challenges of elastic expansion of global data center resources and demand for cross-regional optimization

In the context of global data centers, changes in user-oriented application demands often trigger the need for horizontal scaling across data centers. Research indicates that user geographical mobility and large-scale planned workload migration (such as smart vehicle movement and traffic rebalancing during peak shopping periods) can lead to cross-regional movement of access points. Both six-hour tracking data within a certain cloud platform and Akki's research confirm this pattern - a surge in regional load during large shopping events triggers traffic redistribution, with up to 50% of Facebook requests being processed in different regions during high-load periods, and changes between regions occurring daily.In a single-data-center environment, memory-intensive applications[5]tend to allocate a large amount of memory resources, but when the working set size does not match the system memory, performance is significantly reduced; meanwhile, CPU utilization fluctuates dramatically over time. For a representative workload with high variability, the CPU utilization is below 50% for most of the time within 7 days, with a peak of 91.27%. Users expect to pay according to actual usage, so they prefer an elastic deployment model that independently scales memory and CPU to cope with dynamic load changes within budget, achieving sustained high throughput and low latency. Cloud service providers also aspire to allocate resources flexibly to accommodate diverse customer needs, improve hardware utilization, and reduce investment costs.However, existing resource allocation and software design patterns based on monolithic servers struggle to adapt to rapidly changing cloud hardware architectures, software updates, and cost control requirements, posing two major challenges: the mismatch between client and master data storage locations leads to cross-regional latency, affecting user experience; when managing data at the shard level, the existing Paxos protocol lacks adaptability and flexibility in the face of dynamic fine-grained data association changes in applications, as a single data leader node cannot migrate independently.

3 Research method

3.1 Consensus protocol and optimization challenges in cross-data center database architecture

Cross-regional distributed databases[6] rely on a consistent log replication mechanism, utilizing consensus protocols such as Paxos to ensure data consistency and reliability. The architecture is divided into two modes: single-master and multi-master. In the single-master architecture, there is only one read-write node (master node) globally, and all write requests are executed here before the logs are synchronized to other nodes. However, when the user and the master node are in different regions, cross-regional request forwarding is required, leading to significant performance degradation due to high round-trip times. In the multi-master architecture, data is logically sharded (such as by key-value ranges), with each region holding the consensus group leadership for a designated data shard. Local data can be modified, while other data is only readable. This reduces access latency through data locality, but data access uncertainties and dynamic changes (such as resource failover, load migration, and user mobility) can easily lead to write conflicts, requiring distributed transactions or two-phase commit coordination, which further degrades performance. To address these challenges, research has proposed solutions such as dynamically migrating data consensus group leadership and optimistic concurrency control. For example, the multi-master optimistic concurrency control protocol integrates data replication and transaction processing to minimize coordination and achieve strong consistency. However, practical applications still face key issues: the need to simultaneously achieve high performance and high availability (distributed

transaction protocols such as two-phase commit incur performance overhead), the current research focus on key-value storage models and the lack of support for general relational databases and complex transactions, and the fact that many applications still use non-deterministic or interactive transactions, and lack a suitable model for quantitative analysis to determine the optimal strategy in case of multi-row data write conflicts. Figure 1 shows a schematic diagram of a multi-master architecture database.

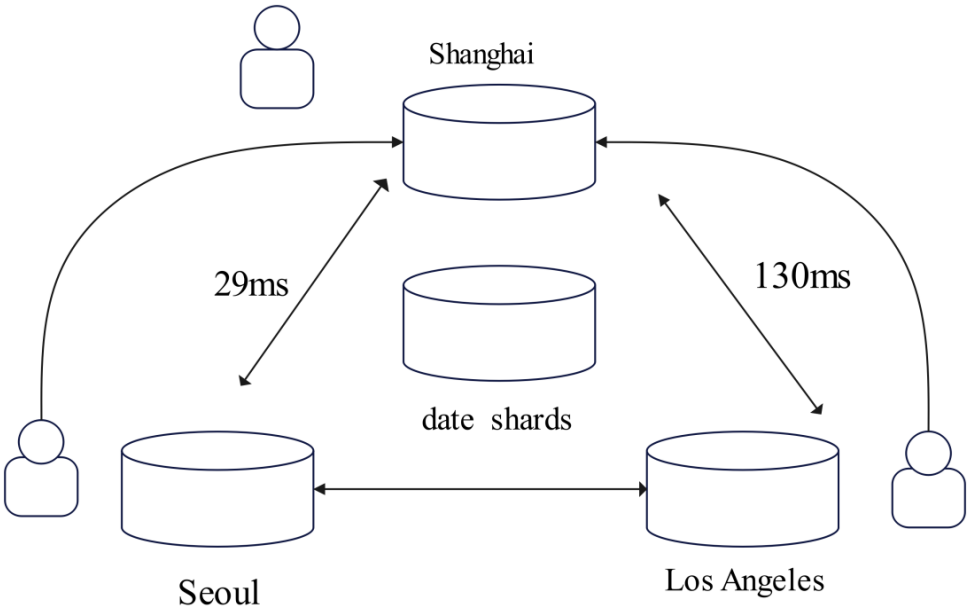

Figure 1 Schematic diagram of a multi-master architecture database

3.2 Research on cloud-native database computing and dynamic memory expansion in a single cloud environment

As applications migrate from local data centers to the cloud, cloud-native relational databases have become a core technology for cloud service providers. Leveraging modern cloud infrastructure, they offer performance comparable to or even superior to traditional databases at lower costs and with higher elasticity. However, the current mainstream cloud-native databases still adopt a standalone/monolithic server architecture, with tightly coupled CPU and memory, making it difficult to meet the ever-growing and highly elastic resource demands of large-scale applications. For instance, analytical queries require a significant amount of memory, potentially exceeding the capacity of a single machine, leading to a significant performance degradation. CPU utilization is mostly low, but during sudden traffic spikes (such as promotional events), it can reach high levels in a short period of time. Static resource allocation can easily lead to resource fragmentation or allocation failures, making it difficult to effectively handle unexpected situations.The existing split architecture[7] (such as InfiniSwap) faces performance bottlenecks when applied in a database environment, mainly stemming from two aspects: each remote page access goes through the complete Linux I/O stack, resulting in network latency that is orders of magnitude higher; general solutions overlook the unique data access patterns of the database kernel, leading to low cache hit rates. Furthermore, the fault tolerance mechanism of the standalone architecture cannot independently handle local and remote memory failures, and time-consuming failure recovery further limits availability. To address these issues, this paper proposes the Legobase architecture, which extends the database kernel buffer pool management module to perceive remote memory by

jointly designing the database kernel and split architecture, bypassing the Linux I/O stack. It leverages optimization opportunities in database workload access patterns, combined with highly optimized caching algorithms (such as an improved LRU mechanism). Simultaneously, it introduces a two-level ARIES protocol to independently handle computing node and remote memory failures—computing nodes and remote memory are unlikely to fail simultaneously. Legobase accelerates the computing node restart process by utilizing data cached in remote memory, optimizing traditional ARIES algorithms through layered dirty page flushing and checkpoint operations. Experimental evaluations show that under TPC-C and TPC-H workloads, even with a large amount of data placed in remote memory, the performance (throughput and latency) of the Legobase system is comparable to that of standalone MySQL settings, and significantly better than MySQL deployed on InfiniSwap (improved by 1.99 to 2.33 times). During failures or planned reconfigurations, Legobase's recovery and warm-up times are 3.87 times and 5.48 times faster than standalone MySQL and InfiniSwap, respectively. The performance advantage stems from the matching of the memory separation architecture with standalone software design (especially traditional fault tolerance mechanisms), avoiding overall memory space recovery and reusing data cached in remote memory. Network latency poses new challenges to the split architecture, affecting data transmission speed, transaction processing determinism, and synchronization consistency. This paper focuses on the impact of the network on data transmission, mitigating network latency through local caching protocols. The core idea is to reduce remote memory access and improve the efficiency of hotspot data caching, while the issue of data synchronization consistency among multiple computing nodes is not deeply explored.

3.3 Implementation and performance optimization analysis of Legobase system

Legobase is implemented based on MySQL, with approximately 10,000 lines of C++ core code and a distributed file system for underlying storage. Its RDMA module constructs an efficient library based on IB Verbs and RDMA CM API, reducing the latency overhead of concurrent requests by maintaining a connection pool. Memory management introduces a two-level LRU implementation, extending the page metadata structure to include remote memory addresses and LRU location fields, supporting both local and remote memory access paths. The ARIES protocol adjusts and intercepts original file system calls, redirecting them to remote memory buffers through RDMA operations, and adds a fast recovery path to accelerate fault recovery. Memory is organized in a hierarchical structure, with LB serving as an RBP cache to meet the rapid expansion/reduction of CPU resource requirements and support load performance improvement beyond the local working set.In terms of performance, Legobase exhibits excellent performance under different memory configurations: even with 50% of memory placed remotely, the throughput and P99 latency in TPC-C and TPC-H tests are still close to the level of standalone MySQL, with a significant improvement of 1.99 to 2.33 times compared to InfiniSwap. Production workload tests show that the sustained high throughput is only 9.96% lower than standalone MySQL, while standalone MySQL's performance under 10% local memory configuration reaches only 39% of its optimal performance. The fault recovery performance is outstanding, with recovery and warm-up times being 3.87 to 5.48 times faster than standalone MySQL and InfiniSwap. In elastic deployment experiments, rapid resource allocation is achieved by dynamically adjusting the number of CPU cores in response to workload changes, resulting in significant cost-effectiveness, saving 44% of monetary costs compared to traditional resource provisioning methods. The overall architecture achieves efficient elastic resource allocation while ensuring data consistency through hierarchical memory organization and protocol optimization.

4 Results and discussion

4.1 PilotDB data replication and state recovery mechanism

The versatility of PilotDB far exceeds the scope of specific Persistent Memory (PM) experimental evaluations. Its design is compatible with multiple Non-Volatile Memory (NVM) technologies [8], such as STT-MRAM, FRAM, ReRAM, PCM, and CXL-based storage/memory technologies (such as XL-FLASH). These technologies provide fine-grained access, ultra-low-latency RDMA compatibility, fast persistence, and large-capacity storage, similar to the functionality of Optane PM. The design remains effective for addressing bandwidth limitations, data persistence, and high CPU load issues that persist across different NVM technologies. As a PM-detached solution for cloud-native relational databases, it has been validated in common database implementations such as MySQL and PostgreSQL. The PMN module can be embedded into a generic detached PM layer, providing cost-effective caching or logging functionality for data-intensive applications based on page-based memory/storage organization. The system implementation includes approximately 5000 lines of C++ code, reusing storage components from distributed file systems. The Compute Node (CN) runs a modified MySQL database[9], mainly optimizing the log module (replacing the original WAL with CDLog and adding a log-pull mechanism) and the buffer pool module (retaining MySQL LRU-managed cache pages and extending the buffer hierarchy to add Remote Buffer Pool RBP). The Persistent Memory Node (PMN) runs a lightweight daemon to handle background log playback and recovery tasks.

4.2 Model experiment

The experiment adopts a heterogeneous cluster testing platform, which includes a 4-node PM layer (each node is equipped with 2 x Intel Xeon Platinum 8260 CPUs, 256GB DDR4 DRAM, 4 x 128GB 3D XPoint Optane DC PM, and 25Gbps network), 8 weakly connected servers (each equipped with 2 processors and 128GB DRAM), and a shared storage layer consisting of 6 servers (providing 3 replicated cloud distributed file system services). The benchmark settings include 100D/0 MySQL deal (all local resources), 10D/100D Legobase (10GB local+100GB remote DRAM cache), 10D/56D Legobase (matching 100GB Optane PM hardware cost), and 10D/100P LegoPM (PM cache replacing remote DRAM cache). The workload uses Sysbench (200GB database, 28 million items, 32 tables), TPC-C (2000 warehouse), and production MySQL workloads, configured with Zipfian 0.99 distribution, with read to write ratios of RO (read-only), RW (7:2), and WO (write only).

In terms of overall performance, the Sysbench results showed that PilotDB achieved a throughput of 97.0% of MySQL iDeal in the WO scenario, which is at least 1.53 times higher than other baselines; The RO scenario accounts for 90.2% of MySQL deal, while the RW scenario has a more significant advantage as concurrency increases. Delay corresponds to throughput, and PilotDB provides similar latency as MySQL iDeal. Table 1 column 64 thread running data: PilotDB has a 26.4%/59.8% higher throughput than LegoPM in RW/WO scenarios, while reducing remote PM write bandwidth consumption by 40.8%/55.5%. Although there is PM write traffic in RO scenarios (due to RBP not hitting page registration), the latency advantage is significant.

Table1 LegoPM vs PilotDB: 200GB DB, 64 Threads Perf & PM Bandwidth

orkload	Throughput (K-QPS)	PM Read (GB/s)	PM Write (GB/s)
RO	153.31 / 153.72	0.63 / 0.63	0.20 / 0.20
RW	100.17 / 126.62	1.16 / 1.10	0.76 / 0.45
WO	90.95 / 145.36	1.05 / 1.00	1.55 / 0.69

In terms of remote CPU consumption, the impact of the number of PM nodes' CPUs is demonstrated: PilotDB-RPC (replacing unilateral RDMA with optimized rRPC) requires 16 cores to achieve peak throughput, which is lower than that of PilotDB but significantly better than LegoPM (which peaks at 8 cores but has a lower peak throughput). LegoPM consumes more PMN CPU resources due to the need for RPC to ensure atomicity for direct remote page writes; PilotDB organizes logs through unilateral RDMA and CN drivers, requiring only 1 core for log writes, thus avoiding PMN CPU becoming a bottleneck.Technical optimization analysis reveals that "Log-offload" only offloads log playback, leading to performance degradation; "CDLog" restores performance through fast and lightweight PM-side log playback, with RW/WO scenarios outperforming LegoPM by 19.7%/62.8%; the complete PilotDB (with log pull) outperforms LegoPM by 26.4%/80.0% in RO/RW/WO scenarios, respectively, and further reduces average/tail latency data consistency. Recovery testing is verified through concurrent workload and fault injection: PilotDB and PostgreSQL run concurrently 50 times without any inconsistency; recovery performance shows that after an 80-second injection crash, PilotDB-CN-crash recovers service in 81 seconds and warms up in 87 seconds; PilotDB-PMN-crash recovery time is significantly better than MySQL-ideal, as RBP cache pages can be reused directly, reducing local LBP filling time.

4.3 Effect analysis

In the Internet service scenario, cross regional database deployment has extended from the initial fault tolerance and disaster recovery to performance optimization and resource management to support the low latency response and load balancing requirements of e-commerce, online games and other applications. Traditional static sharded databases face challenges of cross domain transfer of access points caused by user mobility and load migration [10], and write intensive workloads are further limited in performance due to consistency latency across distant replicas. Existing solutions such as Multi Paxos and Raft support dynamic leadership, but follow traditional state machine replication models, lack multi master coordination mechanisms, and suffer from insufficient flexibility due to coarse-grained shard binding.The PolyBase architecture achieves breakthroughs through row level Paxos group dynamic allocation and recoating mechanisms: decoupling data sharding from consensus group binding, embedding row level metadata directly to support fine-grained locality annotation, dynamically binding data rows to the nearest region leader, and adapting to user mobility and load balancing needs. This architecture integrates the recolor protocol and Paxos logs to ensure consistency across long-distance replicas, while optimizing transaction execution through access heat and correlation mining algorithms to reduce wide area round-trip latency. In cross continent six node production environments and benchmark tests such as TPC-C and YCSB+T, compared to open source and commercial baselines, its throughput increased by 1.5 to 6.9 times, and tail latency decreased by 48% to 86%.The architecture is compatible with the general transaction model, supports key value storage and relational database integration, adapts to consensus protocols such as Raft, and further reduces latency through batch processing and merging of recolored requests. In scenarios of traffic migration, user mobility, and cross continental load balancing, this architecture significantly improves throughput and reduces latency, providing an efficient, flexible, and scalable solution for building cloud native high-performance service engineering systems for real-time decision-making platforms.

5 Conclusion

Cloud native database systems rely on resource elasticity, high availability, and concise interfaces, built on massive resources of cloud platforms, and are suitable for data intensive applications such as e-commerce, financial transactions, and intelligent driving. However, they face

multiple challenges under complex dynamic loads on the cloud: traditional databases have a mismatch between computing and memory resources in data centers, and fail to fully utilize the performance advantages of new hardware; In cross regional scenarios, data consistency and performance are limited by traditional architectures. To build a global service, efficient resource utilization, and high-performance database, this article proposes three innovative architectures. LegoBase, a separated memory cloud native architecture, addresses the issue of insufficient scalability caused by tight coupling between traditional monolithic architecture computing and memory. By extending the relational database buffer pool management system, it effectively utilizes remote memory and bypasses the Linux I/O stack to eliminate efficiency bottlenecks. It introduces a two-layer ARIES recovery protocol to enhance fault tolerance. Experiments have shown that it significantly improves throughput, reduces latency, shortens recovery time, and enhances the resilience of cloud native relational databases. The Persistent Memory Separation Architecture (PilotDB) addresses the issues of write bandwidth limitations and low CPU utilization in persistent memory (PM) separation architecture. It adopts a compute node driven logging mechanism to optimize the process of refreshing data to remote PM, reduce PM node CPU usage, support efficient RDMA friendly remote PM access, reduce bandwidth consumption, and improve overall performance and scalability. Standard SQL benchmark testing and actual production load verification show that it is significantly better than existing solutions in terms of performance, recovery speed, and cost-effectiveness. PolyBase, a cross regional high-performance database architecture, innovatively uses multiple Paxos groups to manage data replicas and consensus group leadership in response to the closely related issues of cross regional transaction performance and execution location. It allows data consensus group leadership to flow freely between multiple regions, and a single row can be independently migrated to the area with the highest access popularity. It also introduces migration strategies to perceive access popularity and data correlation, intelligently decide the most suitable area for data, and the prototype based on MySQL/RocksDB has been tested under typical loads such as TPC-C to significantly improve cross regional distributed transaction performance, flexibly adapt to changes in upper layer application traffic, and achieve a balance between performance and data consistency. Future research directions include optimizing the separated architecture based on the latest network hardware (such as CXL), achieving complete decoupling of computing and storage resources, and maximizing the potential of CXL in resource sharing, low latency transmission, and high-throughput processing; Optimization of database systems for AI applications (such as large language models), improvement of indexing and query optimization techniques to support large-scale concurrent data access, optimization of data flow dynamic adjustment and caching mechanisms; Optimizing data management under multimodal AI training, designing a flexible storage architecture that supports multiple types of data (text, images, audio), combining advanced compression and distributed storage technologies to reduce costs and improve transmission and retrieval speeds, and developing efficient distributed data management and parallel processing technologies to meet the rapid processing needs of large-scale datasets. These innovations together provide an efficient, flexible, and scalable solution for building cloud native high-performance service engineering systems for real-time decision-making platforms.

References

[1] *Zhong S, Rigger M. Scaling Automated Database System Testing[J]. 2025.*

[2] *Zu E, Shu M H, Huang J C, et al. Management Problems of Modern Logistics Information System Based on Data Mining[J]. Mob. Inf. Syst. 2021, 2021:5241921:1-5241921:9. DOI:10. 1155/2021/5241921.*

[3] Liu H, Zhu F, Cheng L. Proof-of-Data: A Consensus Protocol for Collaborative Intelligence[J]. 2025.
[4] Ma, X. (2026). Research on End-To-End Reliability Modeling and Optimization of Service Grid.
[5] Zelin Wang. Data Analysis and Risk in Supply Chain Management. International Journal of Social Sciences and Economic Management (2026), Vol. 7, Issue 1: 132-140.
[6] Weiyao Ma. Automated Operation Approach for Scalable Cloud Data Platform. International Journal of Big Data Intelligent Technology (2026), Vol. 7, Issue 1: 131-139.
[7] Zinuo Wang. Value Reassessment Logic of Resource-Based Enterprises in the Context of Energy Transition. International Journal of Social Sciences and Economic Management (2026), Vol. 7, Issue 1: 141-149.
[8] Xiao Ma. Engineering Study of Disaster Recovery and Fault Self-Healing Mechanisms for Distributed Systems under Cross-Regional Deployment Conditions. International Journal of Engineering Technology and Construction (2026), Vol. 7, Issue 1: 1-7.
[9] Zhixian Zhang. Research on Model Engineering Integration Methods for AI Systems Based on Data-Driven Intelligence. International Journal of Big Data Intelligent Technology (2026), Vol. 7, Issue 1: 140-149.
[10] Zheng, H. (2026). Research on Edge Computing Network Task Scheduling and Resource Management Optimization Based on Artificial Intelligence Technology.
[11] Zheng, H. (2026). Research on Edge Computing Deep Neural Network Task Unloading Based on Resource Collaboration Framework and Multi Strategy Optimization.
[12] Zhang, Z. (2026). Research on the Design of Scalable Enterprise-Level AI Systems Data Platform Architectures from an SDE Perspective.
[13] Yu, X. (2026). Exploration of Multi-Channel Conversion Path Optimization Methods Based on A/B Testing.
[14] Han, X. (2026). Research on Automotive Manufacturing Process Optimization Methods for Multi-Supplier Collaboration.
[15] Hou, Y. (2026). Research on BIOS and BMC Compatibility Optimization Methods for Cross-Generation Servers in Production Environments.
[16] Yin, J. (2026). Research on Financial Time Series Prediction and Multiscale Correlation Based on the Fusion of Network Big Data and Deep Learning.
[17] Yu, X. (2026). Strategy Models and Practical Research of Growth Marketing under the Background of Digital Transformation.
[18] Yu, X. (2025). Digital Transformation Empowers Growth Marketing with Marketing Data Analysis Integration and Real-Time Display Strategy.
[19] Liu, H. (2026). Research on the Application of Causal Reasoning Method in Content Compliance Experimental Evaluation.
[20] Wang, Y. (2026). Research on the Application of Artificial Intelligence in Supply Chain Risk Early Warning.
[21] Sun, Q. (2026). Research on a Robotic Natural Language Intelligent Decision-Making Framework Based on Large Language Models, Thinking Chain Reasoning, and Multi-Agent Collaboration.
[22] Zhou, Y. (2024, November). Construction of a Multi-factor Quantitative Stock Selection System for the New Energy Industry Based on Microservices Architecture and Machine Learning Components. In International Conference on Cognitive based Information Processing and Applications (pp. 163-174). Singapore: Springer Nature Singapore.
[23] Huang, J. (2025, September). Performance Evaluation Index System and Engineering Best Practice of Production-Level Time Series Machine Learning System. In 2025 International

Conference on Intelligent Communication Networks and Computational Techniques (ICICNCT) (pp. 01-07). IEEE.

[24] Hua, X. (2024, November). Design and Implementation of a Game QoE Monitoring and Evaluation System Driven by Network Traffic Analysis. In International Conference on Cognitive based Information Processing and Applications (pp. 149-161). Singapore: Springer Nature Singapore.

[25] Wu, W. (2025, June). Construction and optimization of intelligent gateway software management platform based on jenkins cluster management under cloud edge integration architecture in industrial internet of things. In International Conference on 6G Communications Networking and Signal Processing (pp. 633-645). Singapore: Springer Nature Singapore.

[26] Wu, L. (2025, December). Design and Application of Automatic Data Set Generation Tool Based on KLEE in Embedded Memory Management Performance Test Framework. In 2025 IEEE 17th International Conference on Computational Intelligence and Communication Networks (CICN) (pp. 1111-1117). IEEE.

[27] Qi, Y. (2025, October). Research on Privacy Protection of AI Models in Big Data Using Differential Privacy Technology. In 2025 2nd International Conference on Software, Systems and Information Technology (SSITCON) (pp. 1-5). IEEE.