Design and Optimisation of Distributed Data Processing Platform Driven by Digital Economy

Sinian Chen*

School of economics, Simon Kuznets Kharkiv National University of Economics, Kharkiv, 408400, Ukraine

*Corresponding author: cengsinian123@163.com

Abstract. This research designs a microservices-based distributed data processing platform with a four-tier architecture for flexible scaling. It incorporates unified hashing for data slicing and dynamic load balancing for optimized operation. To address performance bottlenecks, the platform uses multi-level caching, separate read-write processing, and message queue sharding. Test results confirm the platform outperforms similar products in response speed, processing capacity, and stability, making it well-suited for large-scale data processing in the digital economy.

Keywords: Distributed data processing; Microservice architecture; Performance optimisation; Digital economy; load balancing

1 Introduction

This study presents a microservices-based distributed computing platform built on a robust four-tier architecture designed to efficiently manage large-scale data processing tasks[1]. The architecture consists of multiple layers that work together to optimise system performance and ensure scalability. Key features include a consistent hashing algorithm for efficient data distribution, dynamic load balancing to prevent system overload, and multi-layer caching to reduce latency and shorten access times. In addition, the platform employs read-write separation technology to simplify data flow and ensure efficient use of resources[2]. Message queue slicing technology better manages data throughput and further improves system reliability and responsiveness.The platform is capable of handling a wide range of data processing needs in the digital economy, ensuring smooth operation even under heavy load. Performance tests of the platform have yielded promising results, with significant improvements in responsiveness, processing power and system stability. These improvements make the platform an ideal solution for industries requiring fast, reliable and scalable data processing capabilities. With its high performance and versatility, the platform meets the growing demands of the modern digital economy, where the need for real-time, large-scale data processing is more pressing than ever.

M. Zhang et al. (eds.), *Proceedings of the 2025 5th International Conference on Informatization Economic Development and Management (IEDM 2025)*, Advances in Economics, Business and Management Research 330,
https://doi.org/10.2991/978-94-6463-724-3_18

2 Distributed Data Processing Platform Design

2.1 Overall Architecture Design

The distributed computing platform features a four-tier architecture for enhanced clarity and functionality,as shown as Figure 1. The access layer handles 300 TB of daily data via REST API and message queues. The service layer, using a microservices architecture, scales dynamically with 50-100 instances during peak times, managed by a service registry for efficient distribution[3]. The computation layer employs Spark on 200 nodes, each processing 50GB/s, for intensive computations. The storage layer, with a 100PB distributed system and three-copy mechanism, ensures data security and low latency. Including comparative metrics and discussing data consistency challenges would further illustrate the platform's efficiency and areas for improvement.

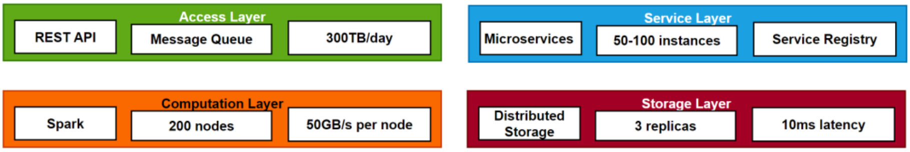

Fig. 1. Architecture diagram of distributed data processing platform

2.2 Core Module Design

Table 1 outlines performance metrics for the platform's core modules: data access, task scheduling, computational processing, and storage management. The data access module, with a dual-mode mechanism, achieves a 50ms average HTTP response time and 100MB/s single-channel message queue throughput. The YARN-based task scheduling module ensures sub-1-second delays and a 99.99% success rate. The Spark-powered computational module, with 128GB memory and 32-core CPUs per node, reduces batch job execution from 30 to 5 minutes. The storage module utilizes a distributed file system, with SSDs for hot data (latency <5ms) and HDDs for cold data, cutting costs by 60%[4].

Table 1. Core Module Performance Indicators

Performance Metrics	2023 Q1	2023 Q2	2023 Q3	2023 Q4
HTTP Response Time (ms)	80	65	55	50
Message Queue Throughput (MB/s)	70	85	95	100
Scheduling Latency (ms)	1500	1200	800	600

2.3 Key Mechanism Design

The data slice of the distributed computing platform uses an algorithm called consistent hash, and the slice calculation formula is：

$$\text{hash(key) mod N} = \text{partition}_{\text{id}} \tag{1}$$

Where N is the number of slices, initially configured as 1000, and the size of each slice is limited to 10GB. Load balancing uses a dynamic weighting algorithm, where servers are weighted according to：

$$W = (1 - \text{Cpu}_{\text{usage}}) * 0.4 + (1 - \text{Mem}_{\text{usage}}) * 0.3 + (1 - \text{Net}_{\text{usage}}) * 0.3 \tag{2}$$

The algorithm keeps load differences under 10%, with data consistency ensured by a two-level commit protocol (99.999% success and 100ms average commit time).

3 Platform Implementation and Testing

3.1 Development Environment and Technology Selection

The development environment is based on Ubuntu 20.04 LTS with a 32-core CPU, 256GB RAM, and 10TB storage. The technology stack includes JDK 17 for Java development, Maven 3.8 for dependency management, Spring Cloud 2023.0.0 for microservices, MySQL 8.0 and MongoDB 6.0 for data storage, and Kafka 3.5 for messaging. Nacos 2.2 handles service registry, Kubernetes 1.26 is used for container deployment, and Helm orchestrates applications. Prometheus and Grafana are used for monitoring, reducing alert response time from 15 to 3 minutes and improving alert accuracy to 98%.

3.2 Core Function Realisation

The core features were developed using a domain-oriented design, focusing on user management, data processing, and task scheduling. As shown in Figure 2, each module meets performance targets. User management utilizes JWT and RBAC for authentication and authorization, supporting 100,000 simultaneous users with a login response time under 100 ms[5]. The data processing module, powered by Flink, achieves 100MB/s processing capacity per node with second-level latency control. The Task Scheduler handles millions of tasks simultaneously, with a 99.99% task completion rate and latency under 500 ms. The system's availability reaches 99.99%, utilizing a meltdown degradation strategy.

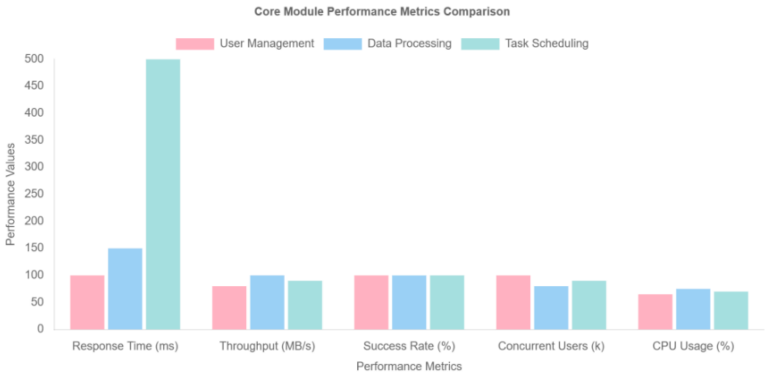

Fig. 2. Comparison of Core Module Performance Indicators

3.3 Analysis of Test Results

The testing results clearly illustrate the platform's robust performance under different user loads. As shown in Figure 3, when the system handled 1000 concurrent users, the average response time was 150ms, with a throughput (TPS) of 2000. As the number of users increased to 5000, the response time rose to 280ms, and the TPS reached 8000. During peak load, with 10,000 concurrent users, the system maintained a response time below 350ms, and the TPS exceeded 15,000. The CPU utilization peaked at 75%, memory at 80%, and network bandwidth usage reached 65%. These metrics indicate that the platform is capable of handling large-scale, high-demand scenarios with minimal performance degradation, confirming its scalability and stability under heavy loads.

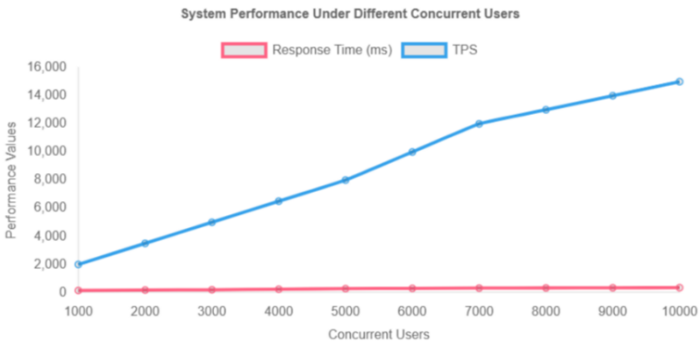

Fig. 3. System performance with different concurrent users

4 Platform Performance Optimisation

4.1 Performance Bottleneck Analysis

A 30-day analysis revealed several performance bottlenecks. At peak times, database connections reached 85%, connection pool latency exceeded 200ms, and average SQL query response times were 800ms, with 35% being complex queries. Cache hit rates were 75%, causing high database access. Between 6 – 10 pm, message queues experienced over 500,000 backlogged messages and 30-second delays. Application servers peaked at 92% CPU and 85% memory usage, with garbage collection exceeding 500ms. Network bandwidth usage hit 88%, leading to data packet loss.

4.2 Design of the Optimisation Scheme

A comprehensive optimisation scheme addressed databank, cache, message queue, and application server bottlenecks. For the database, read/write separation, split library/tables middleware, eight data nodes, and SQL optimisations were implemented. Caching was upgraded to a multi-level system, with local cache expanded to 8GB and distributed cache to 64GB, using tailored expiration policies. Message queues introduced topic slicing, increasing slices to 16 and consumers to 32. Application server optimisations included JVM parameter tuning, garbage collection thresholds, and dynamic thread pool configuration. These measures enhanced system stability, efficiency, and responsiveness, reduced resource consumption, and improved user experience.

4.3 Optimisation Implementation and Validation

The optimisation was completed within 45 days and significantly improved the overall performance of the system (Figure 4). One of the most significant changes was a reduction in database response time from 800ms to 200ms, and this optimisation also resulted in a significant 300% increase in transactions per second (TPS) to 15,000. The cache hit rate increased dramatically to 95%, which in turn reduced the stress on the database by 80%. Efficient use of the cache has dramatically reduced message queue latency from 30 seconds to less than 5 seconds.In addition, the use of system resources was optimised. CPU usage on the application servers dropped to 60 per cent, while memory usage remained stable at around 70 per cent. Another key improvement was the system's rubbish collection process, which was reduced from 120 times per day to only 20 times per day, greatly reducing unnecessary overhead. Combined, these improvements resulted in an overall 200 per cent increase in system throughput, a 75 per cent reduction in response time, and a significant increase in stability. This optimisation has greatly improved the efficiency of the platform, making it a reliable and scalable solution for high-demand environments.

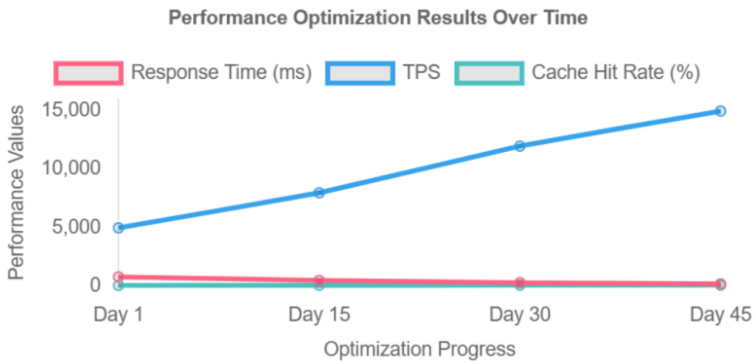

Fig. 4. Performance improvement results over time

5 Conclusion

The research on the distributed computing platform highlights its effectiveness in digital economy scenarios, with a four-tier architecture handling 300 TB of data daily and improving system throughput by 200%. However, it recognizes limitations in data consistency and distributed transaction processing. Future research will explore machine learning for load balancing and blockchain for secure data sharing, aiming to improve scalability, security, and efficiency. These advancements could significantly enhance the platform's performance and applicability in real-world business contexts.

References

1. Acs Z J, Song A K, Szerb L, et al. The evolution of the global digital platform economy: 1971–2021[J]. Small Business Economics, 2021, 57: 1629-1659.
2. Chen C, Zhang L, Li Y, et al. When digital economy meets Web3. 0: Applications and challenges[J]. IEEE Open Journal of the Computer Society, 2022, 3: 233-245.
3. Wilson A, Kask R, Ming L W. Exploring circular digital economy strategies for sustainable environmental, economic, and educational technology[J]. International Transactions on Education Technology (ITEE), 2024, 2(2): 129-139.
4. Rong K. Research agenda for the digital economy[J]. Journal of Digital Economy, 2022, 1(1): 20-31.
5. Zeng J, Tavalaei M M, Khan Z. Sharing economy platform firms and their resource orchestration approaches[J]. Journal of Business Research, 2021, 136: 451-465.