GEORGE BROWN PRESS
JOURNALS, BOOKS & PROCEEDINGS

Economics and Management Innovation

Article

Research on Architecture Optimization of Intelligent Cloud Platform and Performance Enhancement of MicroServices

Xiang Chen [1,*]

[1] Azure, Microsoft, Washington, 98052, USA
* Correspondence: Xiang Chen, Azure, Microsoft, Washington, 98052, USA

Abstract: Intelligent cloud platforms have become a cornerstone in managing and operating complex business process systems. However, the microservice architectures underlying these platforms often face significant challenges in real-world applications, including tight coupling between services, intricate inter-service communication, inefficient resource scheduling, and difficulties in tracing and diagnosing service calls. To address these issues, this study proposes a comprehensive framework that integrates multiple key strategies: architectural decoupling to reduce interdependencies, advanced service governance for standardized and reliable service management, optimized resource scheduling to improve system efficiency, and end-to-end call chain diagnosis to enhance fault detection and performance monitoring. Based on these strategies, a high-performance intelligent cloud platform model is established, capable of achieving both operational efficiency and system reliability. The findings of this research not only provide a solid theoretical foundation but also offer practical guidelines for ensuring the stable operation, proactive maintenance, and intelligent management of microservice-based cloud platforms, ultimately supporting scalable and resilient enterprise IT infrastructures.

Keywords: intelligent cloud platform; microservice architecture; performance optimization

1. Introduction

With the rapid development of cloud computing and artificial intelligence (AI), intelligent cloud platforms have emerged as a critical digital backbone for modern enterprises. These platforms enable organizations to efficiently manage complex business processes, support dynamic workloads, and implement AI-driven decision-making. At the core of these platforms lies microservices architecture, which offers inherent flexibility, modularity, and strong scalability. Despite these advantages, practical implementations often reveal significant challenges, including high architectural coupling, complex inter-service communication, and limitations in system performance under heavy workloads. To address these issues, it is essential to conduct systematic improvements in system architecture design and service scheduling optimization. This article provides an in-depth analysis of the challenges associated with microservice-based intelligent cloud platforms and proposes a set of targeted optimization strategies aimed at improving operational efficiency, service reliability, and overall system scalability.

2. Overview of the Technical Architecture of Intelligent Cloud Platforms

Intelligent cloud platforms build upon traditional cloud architectures by integrating artificial intelligence, big data processing, and automated operation and maintenance technologies, thereby enhancing intelligent resource scheduling, service orchestration, and system resilience. Conceptually, the technical architecture can be divided into four primary layers:

Published: 21 October 2025

1) Infrastructure Layer: This layer leverages virtualization technologies and container orchestration platforms, such as Kubernetes, to abstract physical resources and enable unified resource scheduling across heterogeneous hardware. It ensures efficient allocation of computing, storage, and networking resources while supporting elastic scalability and high availability [1].
2) Platform Support Layer: The platform layer provides essential foundational services, including microservice frameworks, messaging middleware, databases, and caching systems. These components support rapid service development, high-throughput data processing, and seamless inter-service communication.
3) Service Governance Layer: Service governance is critical for ensuring reliability, observability, and maintainability. This layer typically incorporates service mesh frameworks, configuration centers, and distributed tracing tools to enable service registration, discovery, call control, and end-to-end observability. These mechanisms facilitate fault detection, performance monitoring, and adaptive traffic management.
4) Business Application Layer: This layer executes specific microservice applications and intelligent algorithm modules tailored to diverse business requirements. It integrates AI-driven analytics, predictive modeling, and domain-specific logic to deliver actionable insights and support decision-making processes.

The overall architecture emphasizes decoupling, resilience, observability, and automation, creating a foundation for a stable, flexible, and self-regulating cloud infrastructure. By combining modular microservices with intelligent resource management, this layered architecture ensures scalability, robustness, and operational efficiency, enabling enterprises to adapt rapidly to evolving business demands and technological advances.

3. Problems in the Architecture of Intelligent Cloud Platforms

3.1. High Architecture Coupling and Lack of Elastic Scalability

At present, most intelligent cloud platforms have implemented service partitioning; however, a significant number of implicit couplings persist in actual operations. Synchronous call chains, shared databases, and centralized configuration centers can create strong interdependencies between modules [2]. Consequently, when a service encounters problems or experiences performance degradation, cascading effects are likely to occur, leading to reduced system recoverability and weakened local replacement capabilities. Furthermore, suboptimal service deployment granularity and the absence of autonomous resource allocation mechanisms limit the elastic scalability of cloud platforms during peak or fluctuating workloads. Without sufficient decoupling and adaptive mechanisms, these platforms struggle to accommodate dynamic operational demands, such as sudden business surges and resource contention, as summarized in Table 1

Table 1. Comparison of the Effects of Architecture Coupling on Elastic Scalability.

Comparing dimensions	High coupling architecture performance	Decoupling optimization architecture performance
Service dependency relationship	Strong synchronous dependency, highly coupled modules	Asynchronous weak dependency, service independent deployment
Fault infectivity	Local anomalies may trigger global cascading failures	Strong fault isolation, local abnormalities do not affect overall operation
Data storage structure	Shared database, severe resource competition	Independent data persistence layer with strong isolation

Deployment and update flexibility	Service binding deployment, modification involves a wide range of implications	Support hot updates for individual services and flexible iterations
Elastic expansion efficiency	Difficult to expand independently, requiring weighted scheduling of the entire resource block	Support on-demand scheduling and single service elastic scaling capacity

3.2. Complex Communication Between Microservices and Difficulty in Tracing Call Chains

In intelligent cloud platform environments, the large number of microservices and their distributed functions require multiple services to be linked in chain processes for inter-service communication. As business processes grow increasingly complex, communication patterns become intertwined, including synchronous calls between services, asynchronous messaging, and API gateway interactions, making it difficult to clearly define the logical boundaries of communication paths. When a link experiences delays or failures, the platform struggles to quickly locate the fault. Moreover, traditional log-based approaches lack a unified service perspective, and call chains often do not have standardized tracking methods, further complicating maintenance and reducing the effectiveness of performance tuning and fault repair, as summarized in Table 2

Table 2. Comparison of microservice communication complexity and call chain tracing capability.

Comparing dimensions	Performance of complex communication systems	Optimized system performance
Communication method between services	Multiple synchronous calls and complex message paths	Asynchronous decoupling, clear path
Call chain transparency	The multi hop call path is unclear and difficult to trace	Unified tracking of call chain, strong visual traceability capability
Fault localization efficiency	Slow identification of faulty links, relying on manual troubleshooting	Support distributed tracking and fast fault location
Log readability	Service logs are fragmented and lack correlation	Unified log standards, supporting cross service log aggregation
Operation and maintenance response efficiency	Long troubleshooting cycle and slow recovery	Automatic warning and alarm traceability to improve response efficiency

3.3. Uneven Container Scheduling and Resource Allocation Make It Difficult to Optimize System Load

In intelligent cloud platforms, container technologies are widely adopted for the deployment and operation of microservices. However, current container scheduling strategies largely rely on static rules or preset thresholds, lacking real-time awareness of node resource status and dynamic service load fluctuations. This can lead to imbalanced resource allocation, with some nodes experiencing underutilization, such as high CPU idleness or low memory usage, while others face resource contention and delayed responses. Such imbalances make it challenging to dynamically maintain overall system load equilibrium. Furthermore, the absence of flexible and adaptive scheduling mechanisms limits the rapid scalability of services during peak workloads, as summarized in Table 3.

Table 3. Comparative Analysis of Container Scheduling and Resource Allocation Mechanisms.

Comparing dimensions	Performance of traditional scheduling methods	Performance after intelligent scheduling optimization
Basis for scheduling decisions	Static configuration, resource threshold setting	Real time monitoring, load forecasting, and priority scheduling
Node resource utilization rate	Easy to experience resource idle or resource competition	Automatically balancing load and improving resource utilization
Hotspot service response capability	Significant delay during peak hours and delayed expansion	Support automatic elastic expansion and contraction, fast and stable response
Service deployment strategy	Manual scheduling, strong randomness in location	Optimize deployment location based on service dependencies and load conditions
Overall system load performance	Unequal use of resources and unstable system	Dynamically scheduling resources to improve platform operational efficiency

4. Optimization Strategy for Intelligent Cloud Platform Architecture

4.1. Service Decoupling and Event Driven Architecture Design

Service decoupling has become a central objective in optimizing the architecture of intelligent cloud platforms, aiming to address the excessive coupling and complex inter-service call relationships inherent in microservices. By redefining service boundaries and appropriately partitioning responsibilities, business logic can be modularized with minimal interdependencies, reducing direct coupling between services. Event-driven architecture (EDA) serves as an effective decoupling strategy, enabling asynchronous communication between services via message middleware. This approach allows each microservice to respond to events independently, avoiding the blocking and long dependency chains associated with synchronous calls.

In an event-driven model, services publish events without requiring knowledge of subscribers, significantly enhancing system flexibility and scalability. Platforms can utilize message queues such as Kafka or RabbitMQ to support high-throughput, reliable event stream processing. When combined with an event bus and event backtracking mechanisms, the system state becomes traceable and capable of automatic compensation, thereby improving overall stability and robustness [3]. The EDA-based decoupling architecture increases microservice autonomy and provides a solid foundation for elastic scalability and the continuous evolution of platform functionality, as illustrated in Figure 1.

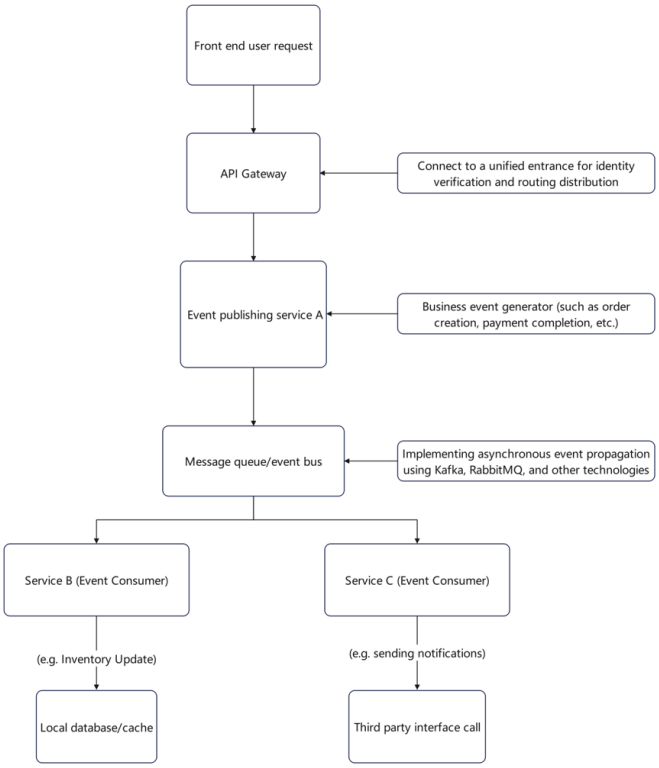

Figure 1. Service Decoupling and Event Driven Architecture Design Framework Diagram.

4.2. Construction of Service Grid and Unified Service Governance Platform

As the number of microservices grows, the communication and management requirements among them become increasingly complex. Traditionally, control logic such as rate limiting, circuit breaking, and authentication has been implemented within business code, which can result in tangled logic and maintenance challenges. Service mesh has emerged to provide unified governance by separating the control plane and data plane. Service-to-service communication is managed by lightweight sidecar proxies, such as Envoy, while governance logic is configured and controlled through service mesh control panels, such as Istio. This approach enhances service autonomy and observability, ensuring that governance policies are consistently enforced across the platform.

In service governance, the overall system availability A can be expressed in terms of the reliability of an individual service R_i and the number of service call chains n as follows:

$$A = \prod_{i=1}^{n} R_i \tag{1}$$

This formula indicates that the failure of any node within a service chain directly impacts the overall system availability. Service mesh mechanisms improve the reliability of individual services R_i through strategies such as circuit breakers, retries, and flow control, thereby enhancing overall system stability. The unified governance platform can further integrate call chain tracking, log collection, and policy management, enabling centralized configuration, real-time monitoring, and automated operation and maintenance. These capabilities provide essential support for intelligent cloud platforms in achieving efficient service governance and secure inter-service communication.

4.3. System Integration of Multi-Level Caching and Asynchronous Mechanisms

The most critical optimization strategy for enhancing the response efficiency and processing performance of intelligent cloud platforms is the combination of multi-level cach-

ing with asynchronous mechanisms. The multi-layer caching architecture typically consists of three levels: local caching, distributed caching, and database caching, aimed at reducing the frequency of database access and improving data retrieval efficiency. Local caches, such as Guava, are suitable for short-lived hotspot data, while distributed caches, such as Redis, are better suited for shared state scenarios, high-frequency access, and other situations requiring consistent data across services.

When handling customer requests, the platform can leverage asynchronous mechanisms, including message queues and task queues, to decouple auxiliary operations-such as logging, email notifications, and audit processing-from core business processes, thereby preventing blocking of critical operations. These asynchronous mechanisms not only increase system throughput but also improve the fault tolerance and operational flexibility of individual modules [4]. By integrating multi-level caching with asynchronous processing, the platform can achieve fast data access, task decoupling, and optimized resource utilization, providing stable support for the high availability and high-performance operation of microservice systems.

5. Performance Improvement Plan for Intelligent Cloud Platform Microservices

5.1. Optimizing Service Scheduling and Resource Elastic Configuration

In intelligent cloud platforms, service scheduling strategies directly influence resource utilization and system responsiveness. Traditional fixed scheduling approaches are insufficient to handle resource pressure arising from fluctuations in service requests, often leading to overutilization of some nodes while others remain idle. To enhance overall system operational efficiency, it is necessary to adopt flexible and intelligent service scheduling strategies. Platforms such as Kubernetes can be leveraged for real-time monitoring and automated decision-making, enabling comprehensive scheduling based on attributes including service load, node resource occupancy, and the geographic distribution of service nodes.

Additionally, the introduction of an auto-scaling mechanism further optimizes resource deployment. By configuring thresholds or employing predictive models, the system can dynamically adjust the number of service instances according to actual application load, ensuring stable operation under high-demand conditions while conserving resources during periods of low activity. The combination of elastic scheduling and intelligent resource configuration mechanisms complements each other, improving system adaptability and cost efficiency, while providing robust underlying operational support for microservice systems, as summarized in Table 4.

Table 4. Comparison of Service Scheduling Strategies and Resource Allocation Methods.

Comparing dimensions	Static scheduling mechanism	Dynamic scheduling and elastic configuration mechanism
Scheduling basis	Fixed rules, manually set	Real time load, resource utilization, service priority
Node resource utilization rate	Unequal allocation of resources can lead to waste or bottlenecks	Automatic balancing significantly improves resource utilization efficiency
Service responsiveness	The response delay is large during peak hours, and the expansion lags behind	Support automatic scaling to ensure stable operation during peak periods
Level of operation and maintenance intervention	Frequent manual intervention and parameter adjustment are required	Highly automated, with adaptive capabilities

Cost control effectiveness	Easy to generate resource redundancy or insufficient configuration	Dynamically control the number of instances and optimize cost expenditures

5.2. Simplified Operating Environment and Execution Architecture Design

The deployment of microservices in intelligent cloud platforms typically relies on container technologies. If the operating environment is redundant, the system base image is large, or runtime resource consumption is high, system startup speed and overall performance can be adversely affected. To enhance system responsiveness, it is necessary to reduce container image size, address dependency issues during runtime, remove redundant components, and construct a more lightweight operating environment. For instance, using streamlined system images such as Alpine can significantly reduce resource overhead and associated risks, improving service startup speed and security.

In addition to these measures, adopting a serverless architecture is another effective approach to simplifying the runtime environment. By executing functions on demand and triggering events, the platform can minimize long-term resource occupation and enhance resource utilization [5]. For tasks that do not require continuous operation or intensive computation, combining container technologies with Function-as-a-Service (FaaS) frameworks, such as Knative, enables minimal deployment scale and instant invocation, thereby improving service flexibility and system performance. Through the simplification of operating environments and architectures, microservice systems can achieve higher execution speed and increased resilience, as summarized in Table 5.

Table 5. Performance and Resource Performance Comparison of Different Operating Architecture Modes.

Comparing dimensions	Traditional container architecture	Lightweight container architecture	Serverless (FaaS) architecture
Mirror volume	Larger (>200MB)	Simplified (<50MB)	No need to deploy images (platform hosting execution)
Start Time	Seconds to tens of seconds	1-3 seconds	Millisecond level cold start (or Warm container trigger)
Resource occupancy	Resident resources with significant fluctuations	Stable occupancy, low	Dynamic calling, with the strongest resource elasticity
Architecture maintenance complexity	There are numerous dependency configurations, making updates difficult	Simplified dependencies, easy to upgrade	Full tube operation, simplest maintenance
Applicable scenarios	Persistent service, complex business processing	High frequency interface, response service	Event triggered and intermittent computational tasks

5.3. Analyzing Call Links and Diagnosing Performance Bottlenecks

In a microservice environment, the high degree of service decomposition enhances system resilience but introduces greater complexity to request paths. A single user request may involve multiple service nodes, asynchronous operations, and external interface calls, which significantly increases the difficulty of problem tracking and performance analysis. To address this, the platform must implement a highly observable call chain tracing system, leveraging distributed tracing tools such as Zipkin, Jaeger, or SkyWalking to achieve transparent monitoring of inter-service call paths.

Call chain analysis enables clear identification of service dependencies and response times, as well as detection of delay bottlenecks, abnormal nodes, and request blocking points. By integrating log and metric monitoring tools such as ELK and Prometheus, the platform can detect anomalies in real time and automatically issue alerts, thereby improving operational efficiency and system stability. Through the combined monitoring of call chains and performance metrics, intelligent cloud platforms can achieve proactive perception, precise fault localization, and fine-grained optimization, establishing a more robust and reliable service system, as summarized in Table 6.

Table 6. Comparison of Mainstream Call Chain Tracking Tool Functions.

Tool Name	Core functions	Visualization ability	Data storage method	Integration difficulty	Applicable scenarios
Zipkin	Call tracing and latency analysis	Basic topology view	Built in storage/Elasticsearch	low	Small/Medium sized Service System
Jaeger	Full link tracking, performance bottleneck localization	Strong visualization and support for aggregation	Elasticsearch	medium	Kubernetes container environment, distributed system
SkyWalking	Service topology, link analysis, and metric monitoring	Strong, supports charts and alerts	Built in+Elastic/MongoDB	tall	Large scale microservice system, operation and maintenance automation scenarios
Pinpoint	Method level call tracing, real-time graph	Rich and supports link flow	HBase and other big data platforms	tall	High concurrency systems and refined application monitoring requirements

6. Conclusion

The stability and performance optimization of intelligent cloud platforms are crucial for advancing enterprise intelligence and supporting complex digital operations. This article has examined the inherent challenges in microservice architectures, including excessive coupling, complex inter-service communication, and resource sharing conflicts, and has proposed a range of solutions. These solutions include decoupling design through event-driven architectures, service grid governance for improved reliability and observability, multi-level caching combined with asynchronous mechanisms, elastic service scheduling, container optimization, and serverless deployment strategies.

Through this research, it is evident that constructing a cloud platform architecture characterized by high reliability, strong scalability, and agile feedback mechanisms can significantly enhance operational efficiency, fault tolerance, and the intelligent management of microservices. Moreover, such an architecture provides a foundation for continuous evolution, enabling the platform to adapt to fluctuating workloads, optimize resource utilization, and maintain high performance under varying operational conditions. Looking forward, the integration of emerging technologies such as AIOps, edge computing, and real-time analytics will further empower intelligent cloud platforms to achieve predictive maintenance, autonomous decision-making, and enhanced service quality, ultimately supporting more resilient and intelligent enterprise ecosystems.

References

1. Z. Huang, H. Zhao, Z. Nan, H. Ma, X. Li, H. Li, and S. Nie, "Application and practice of industrial IoT cloud platform in oilfield intelligent transformation," In Journal of Physics: Conference Series, October, 2024, p. 012004. doi: 10.1088/1742-6596/2874/1/012004.
2. L. Borsoi, E. Listorti, and O. Ciani, ", & Cinderella Consortium," (2024). Artificial-Intelligence Cloud-Based Platform to Support Shared Decision-Making in the Locoregional Treatment of Breast Cancer: Protocol for a Multidimensional Evaluation Embedded in the CINDERELLA Clinical Trial. PharmacoEconomics-Open, vol. 8, no. 6, pp. 945-959, 2024.
3. A. Morchid, M. Marhoun, R. El Alami, and B. Boukili, "Intelligent detection for sustainable agriculture: A review of IoT-based embedded systems, cloud platforms, DL, and ML for plant disease detection," Multimedia Tools and Applications, vol. 83, no. 28, pp. 70961-71000, 2024. doi: 10.1007/s11042-024-18392-9.
4. F. Rodrigues, F. Pinelas, S. Ferreira, M. Rodrigues, and N. Rocha, "A Recommendation System Based on a Microservice Architecture to Avoid Workplace Stress," Electronics, vol. 14, no. 7, p. 1446, 2025. doi: 10.3390/electronics14071446.
5. H. J. Choi, J. H. Kim, J. H. Lee, J. Y. Han, and W. S. Kim, "Adaptive Microservice Architecture and Service Orchestration Considering Resource Balance to Support Multi-User Cloud VR," Electronics (2079-9292), vol. 14, no. 7, 2025.