Scholar Publishing Group
Machine Learning Theory and Practice
https://doi.org/10.38007/ML.2026.060109
ISSN 2790-0983 Vol. 6, Issue 1: 77-84

Performance Evaluation of Distributed Transaction Consistency Protocols for Microservice Architectures in Financial Systems —Focusing on 2PC/XA, TCC, and Saga Protocols

Jianheng Lu

Guangzhou Huashang College, Guangdong Guangzhou, 511300, China

Keywords: Financial microservices; Distributed transactions; Consistency protocols; Performance evaluation; Saga; TCC

Abstract: With the increasing decomposition of core financial accounting, payment clearing, and risk control into microservices, the security, throughput, and auditability of cross-service transaction consistency have become increasingly important. This paper analyzes recent English literature and publicly available experimental data, comparing the performance of three mainstream protocols—2PC/XA, TCC, and Saga—in five aspects: link latency, throughput, failure retries, compensation costs, and observability overhead. The results show that 2PC/XA is suitable for short-link, high-constraint, and strongly consistent scenarios; TCC is more suitable for business processes that can be abstracted into intermediate states, such as account freezing and credit limit reservation; and Saga is more suitable for cross-domain, long-link, and consistency transaction processing flows that allow for eventual consistency. Therefore, this paper proposes a path for transaction layered governance and performance optimization from the perspective of financial systems.

1. Introduction

The digital transformation of the financial system is not simply about moving the system to the cloud, but about redefining the boundaries of business, data, and responsibility. Traditional monolithic core systems rely on single-database transactions to ensure ACID semantics. After payment, clearing, credit granting, credit limits, risk control, and notification are split into multiple autonomous services, what was originally a single commit action completed within the database has become a complex process that requires collaboration across networks, instances, and storage engines. For the financial industry, this change is even more pronounced. On the one hand, transaction records, balance changes, and accounting flows need to have strong correctness and traceability; on the other hand, the matching, payment, and risk control links need to achieve very low latency and high throughput. Any additional

coordination cost will be amplified during peak periods. Research on microservice transaction management has expanded from the traditional 2PC/XA to TCC, Saga, multi-storage unified transaction managers, transaction observability, and causal analysis[1]-[6].

There are still two major contradictions in existing engineering practices. The first is the contradiction between consistency and performance. Strong consistency protocols use locks and coordination to ensure correctness, but they can cause blocking and cascading timeouts. The second is the contradiction between business abstraction and implementation complexity. Compensation protocols improve scalability, but they leave rollback semantics to business developers, increasing the design cost of idempotency, retry, and auditing logic [2][6]. Therefore, evaluating distributed transaction consistency protocols from the perspective of financial business characteristics and obtaining applicable selection criteria has become an important issue in microservice architecture governance.

2. Analysis of the Current Research Status

In the past three years, research on distributed transactions in microservices has mainly focused on four aspects. First, continuous transformation is carried out on the basis of traditional transaction management so that it can maintain strong isolation in multi-database or heterogeneous storage environments. ScalarDB, Epoxy, etc. have moved the transaction boundary upward with a unified transaction abstraction or authentication method, thereby providing users with consistent access semantics in a multi-data source environment [3][5]. Second, research is conducted on the reconstruction of applications and the splitting of transactions. The transaction boundary and service boundary of monolithic applications are analyzed in an automated way to reduce the risk of transaction distortion when migrating monolithic applications to microservices. The third point is to emphasize event-driven and asynchronous transaction platforms, and to achieve eventual consistency by using transaction logs, message boxes and orchestration mechanisms with low coupling. Research is carried out around the performance analysis and observability of microservices, using causal inference, distributed tracing and runtime analysis tools to explain the source of tail latency and provide evidence for protocol-level tuning [7] to [10].

The publicly available results show that strong consistency protocols still maintain a stable advantage in short-link, low-conflict, and controllable network environments. However, when the number of participants increases, the number of network round trips increases, and the lock holding time becomes longer, the system's tail latency increases significantly. Experiments by Bashtovyi and Fechan show that after changing the business from monolithic transactions to microservice distributed transactions, at TPS of 100, 500, and 1000, the average latency increased from the original 224ms, 255ms, and 411ms to 876ms, 3243ms, and 5117ms, respectively. Queirós' research found that in highly competitive environments, reasonable consistency control can significantly reduce the overall latency caused by repeated interruptions in read operations, resulting in a 2.63-fold improvement in latency at 640 req/s.

3. Problem Statement

Although there are various solutions such as 2PC/XA, TCC, Saga and Outbox in academia and industry, there are still four practical problems when they are directly applied to financial microservice systems. First, the semantics of the protocol do not fully correspond to the semantics of financial

business. The technical commit and rollback cannot fully correspond to business operations such as account freezing, credit limit release, and order reversal. Second, the performance fluctuation is underestimated during peak traffic. Financial systems often exhibit pulse-like concurrency during payroll, promotion and settlement. If the coordinator, message broker and transaction log have bottleneck problems, it will increase the jitter of the link. Third, there is a lack of quantifiable data support. When transaction participants, compensation chains and retry chains cross many service instances, it is difficult for engineers to find the root cause of performance degradation without good tracing and causal analysis capabilities [7][8]. Fourth, the lack of a layered standard for protocol selection leads to extreme cases in the design of global strong consistency or global eventual consistency, resulting in high costs and low system resilience.

Therefore, this paper abstracts the problem into a unified evaluation system that considers latency, throughput, success rate, compensation cost, and governance complexity in a financial microservice environment, and then determines the applicable scope of the transaction consistency protocol based on this system.

4. Problem Solving and Strategies

(I) Establishing a unified performance evaluation index system. To avoid the limitations of evaluating transaction protocols solely using average response time, this paper breaks down microservice transaction performance into five indicators: link latency, commit success probability, coordination amplification factor, compensation recovery cost, and observable overhead. The end-to-end link time can be written as:

$$T_{end} = T_{net} + T_{lock} + T_{log} + T_{retry} \tag{1}$$

In the formula, T_{end} represents the time required for a financial transaction from request to final confirmation; T_{net} is the communication time between services; T_{lock} is the resource lock waiting and overhead time; T_{log} is the cost time for transaction log, message persistence, and tracking reporting; and T_{retry} is the time consumed due to retries or compensation.

Submission success rate can be expressed as:

$$S = N_{committed} / N_{total} \tag{2}$$

$N_{committed}$ is the number of transactions that were committed and confirmed within the observation window, and N_{total} is the total number of transactions that entered the system.

Throughput is defined as:

$$TPS = N_{committed} / \Delta t \tag{3}$$

In the formula, Δt represents the statistical window. High throughput does not necessarily mean high efficiency. If a large number of retries or compensations are used to increase apparent throughput, the system's true effective throughput will decrease.

Considering that both the number of coordination rounds and the number of participants can amplify latency, an approximate function for the coordination overhead is given as follows:

$$C_{coord} \approx P \times R \times d + \delta \tag{4}$$

Where P is the number of participants in the transaction, R is the number of coordination rounds of the transaction, d is the average propagation latency across services in a single transaction, and δ is the

overhead of log writing to disk plus state synchronization.

To facilitate protocol selection, a comprehensive evaluation score is provided:

$$Score = w1(1-L\_norm) + w2S\_norm + w3(1-C\_norm) + w4(1-O\_norm) \quad (5)$$

In the formula, L_norm, S_norm, C_norm and O_norm represent the standardized latency, success rate, compensation cost and observation overhead, respectively; w1-w4 are the weights configured according to the importance of the business.

(II) Design Protocol Comparison and Layered Selection Method. 2PC/XA is suitable for scenarios with a limited number of services, resource manager support for XA and must achieve atomic commit, i.e. strong consistency write scenarios such as core ledger and end-of-day batch processing. TCC decomposes transactions into Try, Confirm, and Cancel, and transfers the lock semantics to the semantics of business reservation, which is suitable for processes with relatively clear intermediate states such as account freezing, quota occupation, and inventory reservation. Saga is more suitable for approval, order, notification, and settlement peripheral processes, and uses local commit and compensation to achieve eventual consistency. It can improve scalability under high concurrency, but it is important to pay attention to compensation idempotency, sequential consistency, and audit integrity [1][4][11].

Table 1 provides a comparison of the characteristics of consistency protocols for financial microservices.

Protocol	Consistency	Blocking	Rollback	Latency Risk	Typical Use
2PC/XA	Strong	High	Automatic	High	Core ledger
TCC	Strong at business layer	Medium	Explicit cancel	Medium	Freeze / reserve
Saga	Eventual	Low	Compensation	Medium-Low	Order workflow
Outbox + Event	Eventual	Low	Replay / Repair	Low	Notification

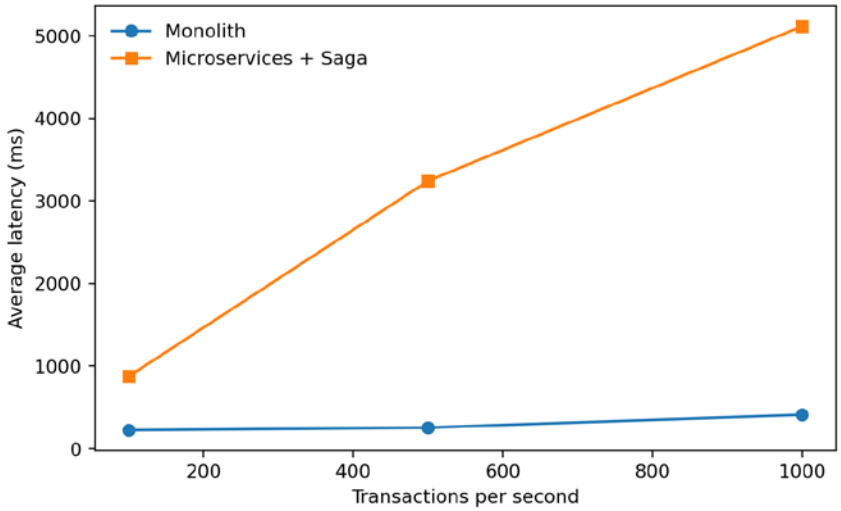

Figure 1 shows a comparison of the average latency of monolithic transactions and microservice

distributed transactions (redrawn using publicly available experimental data).

(III) Analysis of performance differences based on publicly available experimental data. Figure 1, redrawn based on publicly available experimental data from Bashtovyi and Fechan, clearly shows that after the business changed from monolithic transactions to Saga-based microservice distributed transactions, the average latency increased non-linearly.

Figure 1 illustrates that at 100 TPS, the transaction latency of microservices is more than four times that of monolithic services, and this gap continues to widen at 1000 TPS. This means that if too many serial participants, message brokers, and state synchronization nodes are added to the financial transaction chain, the round-trip time at the protocol layer will quickly become a bottleneck.

Figure 2 is drawn based on publicly available graphical data from Queirós, showing the difference in 95th percentile latency between the μTCC with improved consistency and the underlying system under different contention conditions in mixed read-write transactions.

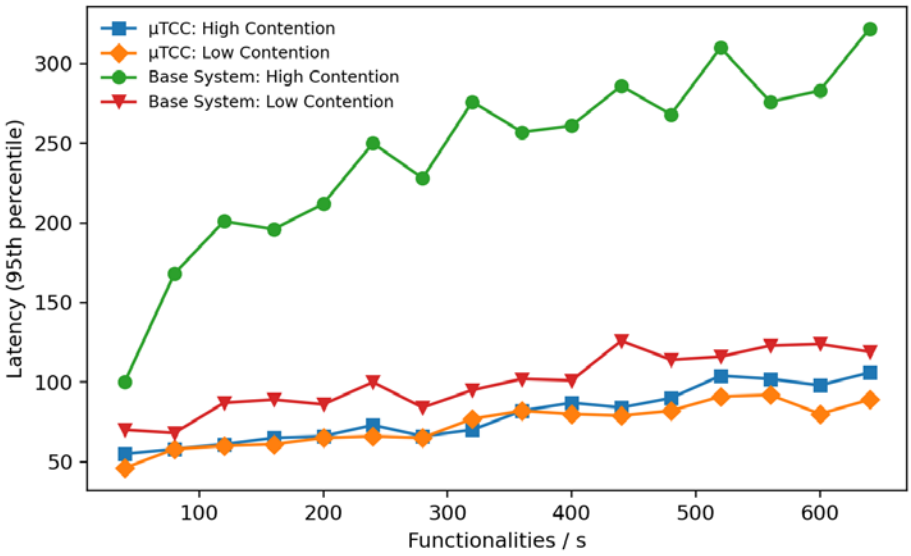

Figure 2 shows a comparison of latency for hybrid transactions (P95) between μTCC and the underlying system (redrawn from publicly available graphical data).

As shown in Figure 2, under high-competition conditions, the tail latency of the basic system increases sharply with the load. μTCC minimizes the number of read transactions that are repeatedly aborted and re-executed, effectively offsetting the tail latency degradation. This result is of reference value for mixed read-write operations such as financial risk control, credit limit verification, and account snapshot reading.

(iv) Provide engineering optimization approaches for financial systems. First, divide transactions into three layers based on business importance: the core accounting layer uses 2PC, XA, or high-constraint TCC; the intermediate transaction layer uses TCC; and the peripheral process layer uses Saga and Outbox. Second, reduce the number of transaction participants by separating read-only queries, notification sending, and non-critical logs from the main transaction chain. Third, create a low-overhead observation system. Nõu et al. argue that distributed tracing leads to a significant decrease in

throughput and an increase in latency; therefore, strategies such as sampling, asynchronous export, and critical path priority tracing are needed, rather than full collection of all links. Fourth, optimize logs and compensation models to ensure that compensation actions are idempotent, traceable, and auditable, defining a compensable, replayable, and reconcilable state machine at the business layer.

Table 2 provides some suggested protocols and their respective focuses based on literature and financial engineering experience.

Scenario	Preferred Protocol	Primary Goal	Main Risk	Control Focus
Account posting	2PC/XA	Strong correctness	Blocking	Short chain
Balance freeze	TCC	Confirmable state	Cancel failure	Idempotency
Order settlement	Saga	Scalability	Compensation drift	State machine
Risk notification	Outbox + Event	High throughput	Delay accumulation	Replay
Cross-domain inquiry	Versioned read	Tail latency	Read abort	Snapshot control

5. Conclusion

Distributed transaction consistency protocols are not the best solution without a business environment. The three solutions, 2PC/XA, TCC, and Saga, correspond to the governance philosophies of strong consistency guarantees, confirmation of intermediate business states, and eventual consistency compensation, respectively. Public research shows that simply migrating a single transaction mechanically to cross-service coordination will result in significant time amplification and load reduction; however, by utilizing more reasonable control of consistency, conflict handling, and transaction layering, better performance resilience can be achieved while meeting financial audit requirements.

This paper establishes an operational performance evaluation system based on the characteristics of financial business. Using literature from the past three years and publicly available experimental data, it summarizes the latency risks, applicable scope, and improvement methods of common protocols. Then, by examining real bank or securities trading logs, a more detailed model of protocol behavior under conditions of multi-site active-active architecture, cross-datacenter network jitter, and regulatory audit pressure can be achieved.

Fungding

2022HSXS087 Research on Object Detection Algorithms Based on Deep Learning

References

[1] *Liu, X., & Yang, D. (2025, March). LLM Data Strategy: Improving Data Availability and Efficiency. In Doctoral Symposium on Computational Intelligence (pp. 425-437). Singapore: Springer Nature Singapore.*

[2] Zhang, Q. (2025, October). Application of Reinforcement Learning in Dynamic Advertising Content Generation. In 2025 2nd International Conference on Software, Systems and Information Technology (SSITCON) (pp. 1-5). IEEE.

[3] Wu, Y. (2025, October). Multi-Level Belief Rule Base Modeling Architecture and Intelligent Optimization Technology for Decision Support Systems. In 2025 2nd International Conference on Software, Systems and Information Technology (SSITCON) (pp. 1-8). IEEE.

[4] 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.

[5] Q. Xu, "Implementation of Intelligent Chatbot Model for Social Media Based on the Combination of Retrieval and Generation," 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Hassan, India, 2025, pp. 1-7, doi: 10.1109/IACIS65746.2025.11210989.

[6] Wu Y. Optimization of Generative AI Intelligent Interaction System Based on Adversarial Attack Defense and Content Controllable Generation[J]. 2025.

[7] Sun J. Quantile Regression Study on the Impact of Investor Sentiment on Financial Credit from the Perspective of Behavioral Finance[J]. 2025.

[8] Wang Y. Application of Data Completion and Full Lifecycle Cost Optimization Integrating Artificial Intelligence in Supply Chain[J]. 2025.

[9] Chen M. Research on Automated Risk Detection Methods in Machine Learning Integrating Privacy Computing[J]. 2025.

[10] 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.

[11] Liu, H. (2026). Research on the Application of Causal Reasoning Method in Content Compliance Experimental Evaluation.

[12] Yu, X. (2026). Strategy Models and Practical Research of Growth Marketing under the Background of Digital Transformation.

[13] Hou, Y. (2026). Research on Server Performance Stability Assurance Mechanisms during Cross-Generation Computing Platform Upgrades.

[14] Han, X. (2026). Research on Automotive Manufacturing Process Optimization Methods for Multi-Supplier Collaboration.

[15] 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.

[16] Hou, Y. (2026). Research on Heterogeneous Server Upgrade Strategies and Resource Utilization Efficiency Oriented Toward Green Computing Objectives. Advances in Computer and Communication, 7(1).

[17] Ding, J. (2025). Exploration of Process Improvement in Automotive Manufacturing Based on Intelligent Production. International Journal of Engineering Advances, 2(2), 17-23.

[18] Lu, Z. (2025). Design and Practice of AI Intelligent Mentor System for DevOps Education. European Journal of Education Science, 1(3), 25-31.

[19] Wu Y. Software Engineering Practice of Microservice Architecture in Full Stack Development: From Architecture Design to Performance Optimization[J]. 2025.

[20] Chen, M. (2026). Research on Privacy-Preserving AI Model Training and Validation Methods Based on Federated Learning.

[21] Yiting Hong. Differentially Private High-Dimensional Business Data Publishing and Analysis Algorithm. International Journal of Business Management and Economics and Trade (2026), Vol. 7, Issue 1: 28-35.

[22] Xu, D. (2026). Analysis of the impact of video infrastructure optimization on large-scale content quality improvement.

[23] Pan, H. (2025, March). Research on Efficient Computing Model of Hartree Fock and Density Functional Theory Based on GPU Acceleration. In Doctoral Symposium on Computational Intelligence (pp. 485-496). Singapore: Springer Nature Singapore.

[24] Zhang, C., Han, J., Zou, Y., Dong, K., Li, Y., Ding, J., & Han, X. (2024, April). Detecting the anomalies in LiDAR pointcloud. In WCX SAE World Congress Experience. SAE Technical Paper.

[25] Wu, Y. (2026). Federated Learning-based Algorithm Design for Privacy Preservation in Cross-domain Data Sharing. Engineering Advances, 6(1).

[26] Sun, J. (2025). Research on Business Data-driven Risk Prediction Methods Based on Machine Learning. Advances in Computer and Communication, 6(4).

[27] Yanchun Wang. (2025) Research on Enhancing ERP System Efficiency Through AI in Cross-border Supply Chain Environments. Advances in Computer and Communication, 6(5), 268-273.

[28] Yanchun Wang. (2025) Research on Enhancing ERP System Efficiency Through AI in Cross-border Supply Chain Environments. Advances in Computer and Communication, 6(5), 268-273.

[29] Zhou, Y. (2026). Energy efficiency and sustainability strategies for data centers. European Journal of Engineering and Technologies, 2(1), 46-53.

[30] Hou, Y. (2026). Exploration of Data Center Cost Optimization Pathways under Multi-generation CPU and GPU Collaborative Architectures. Engineering Advances, 6(1).