

Design and Implementation of High Availability in Distributed Payment Systems Based on Microservices Architecture

Kaiyu Liao

School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, 97331, Oregon, USA

Keywords: Microservice Architecture; Distributed Payment; High Availability; Gray Failure; Transaction Recovery

Abstract: This study addresses the issue in microservice distributed payment systems where Payment replicas remain infrastructure-healthy, but transaction chains have experienced gray-faults, and static retries may amplify anomalies and cause duplicate unconfirmed payment commits. To resolve this, a Transaction Aware Adaptive High Availability Recovery (TA-AHR) mechanism is proposed. This mechanism uses error rate, timeout rate, rolling p99, and transaction progress age to identify instance degradation and achieves secure recovery through endpoint isolation, global retry budgeting, and transaction verification. Controlled fault injection experiments were conducted using OpenTelemetry Demo 3.0.0. Results under continuous latency and intermittent packet loss conditions show that TA-AHR achieves detection success rates of 84.7% and 75.0%, respectively, representing improvements of 19.4 and 25.0 percentage points compared to the static resilience baseline, and SLO effective availability improvements of 1.7 and 3.8 percentage points. The p99 in normal conditions increases by 8.7% compared to the baseline. No duplicate valid commits were observed during transaction verification, but 2.25% of uncertain transactions required deferred verification. The study demonstrates that when healthy redundancy exists, TA-AHR can improve payment gray-fault recovery while balancing service availability and transaction correctness.

1 Introduction

Microservice architecture reduces the direct coupling between modules within a large software system by splitting business functions, deploying them independently, and scaling them elastically, enabling services to evolve independently according to business boundaries [1]. In a payment system, order confirmation, payment execution, and accounting can be deployed separately, but the complete transaction also spans more service instances and network boundaries. As the dependency chain lengthens, system anomalies are no longer just manifested as process exits; local network latency, intermittent packet loss, and single-instance performance degradation can all be transmitted

along the payment chain and affect the final business state.

Cloud-native platforms such as Kubernetes typically rely on multiple replicas, health checks, and instance restarts to maintain service continuity. These mechanisms are effective for explicit failures such as Pod exits, but the infrastructure status cannot fully explain whether a transaction can be completed within the business timeframe. Existing research on fault injection has found that when network latency and runtime load increase simultaneously, cloud-native applications often experience tail latency and throughput degradation first, and only after the fault worsens do obvious request failures occur [2]. This means that while the Payment is still in the Ready state, the Checkout-Payment path may have already lost its actual payment capability.

Fixed timeouts, retries, and circuit breakers can limit some remote call exceptions, but controlled experiments show that the effects of retry and circuit breakers vary with workload and failure rate, and may also cause additional performance loss under continuous failures [3]. Payment scenarios are also subject to transaction semantics constraints. After microservices are split, a business transaction often spans multiple independent services and data states [4], and service recovery does not mean that the original transaction result has been determined; existing research also shows that inconsistencies in execution states across microservices need to be identified and handled separately [5]. For example, when a payment has been submitted but the response is lost, checkout can only see the timeout. If the recovery logic retries directly, the original service recovery action may be transformed into a second valid payment.

Given this contradiction, this study does not extend to general microservice root cause diagnosis, but instead focuses on the safe recovery from gray-fault failures of a single Payment replica in the Checkout-Payment critical link. This study proposes a Transaction-Aware Adaptive High-Availability Recovery (TA-AHR) mechanism, which addresses the key question: when a Payment replica is still considered healthy by the infrastructure, but its actual transaction capability has degraded, can transaction-level runtime status improve the identification of abnormal replicas and the availability of payment services, and prevent unconfirmed transactions from forming duplicate valid commits during replica switching? Continuous latency and intermittent packet loss serve as core experiments; normal state, Pod exit, no healthy replica, temporary unavailability of transaction state storage, and progressive latency are only used to verify overhead, boundaries, and robustness, ensuring that the evidence throughout the study consistently revolves around the same recovery problem.

2 Research Methods

2.1 Research Design and Experimental Subjects

This study employs controlled, repeated fault injection experiments, using the OpenTelemetry Demo 3.0.0 as the experimental prototype. The research focuses solely on the Checkout-Payment critical payment path. Checkout handles settlement orchestration, while Payment processes payment requests and generates transaction results. Existing services such as Cart and Accounting continue to run but are not involved in TA-AHR fault determination. Payment is deployed with two replicas, and all methods utilize the same resources. This configuration constitutes minimum redundancy: after one replica degrades, only one healthy replica remains to handle traffic, facilitating the observation of the relationship between isolation benefits and remaining capacity. Additional replicas are not within the scope of this study.

TA-AHR is implemented directly in the Checkout-side payment call module without adding a separate microservice. This module reads the Payment replica list through Kubernetes End-pointSlice and maintains an independent gRPC client and running status for each endpoint.

$$\text{State}(P_i) \in \{\text{HEALTHY}, \text{DEGRADED}, \text{RECOVERING}\}.$$

B0, B1, and TA-AHR share the same endpoint discovery mechanism and client pool. B0 selects replicas solely based on the Kubernetes Ready state; B1 adds static timeouts, retries, and circuit breakers; TA-AHR can mark replicas as DEGRADED based on business status even when they are still Ready. This unified routing foundation reduces comparison bias.

A single payment transaction is represented as :

$$\tau_i = (id_i, k_i, t_i, s_i),$$

Here, id_i is the transaction identifier, k_i is the idempotency key, t_i is the start time, and s_i is the transaction state, which can be INIT, PROCESSING, SUCCEEDED, or FAILED. The experiment adds a PostgreSQL 16.3 transaction status table, establishing a unique constraint with idempotency_key, and using conditional updates to maintain atomicity for state transitions. F5 only studies transaction uncertainties where "the SUCCEEDED state has been reliably persisted, and the subsequent response is lost," and does not include atomic windows where payment side effects have been committed but the state has not yet been persisted in the conclusions of this study.

Table 1 Experimental environment and freezing parameters

project	set up
Application Prototype	OpenTelemetry Demo 3.0.0
Container orchestration	Kubernetes 1.33.x
Fault Injection	Chaos Mesh 2.8.3
Load tools	k6, checkout-only workload
Payment copy	2
Transaction state storage	PostgreSQL 16.3
Total resources of experimental nodes	Single-node Linux x86-64, 8 vCPUs, 16 GB RAM
Environment Snapshot	CPU model, kernel, container runtime, k6 version, and commit SHA are written to run_manifest.
Detection refresh cycle	1 s
Tail Delay Aggregation Window	30 seconds, minimum 50 samples
Gray fault judgment window	10 s, 6/10 outliers
Payment delay SLO	800 ms
Global retry budget	Within a 10-second window, $\text{Retry}/\text{Original} \leq 0.25$
Single transaction retry limit	2 times, 250/500 ms retreat
Core experimental sample size	72 times per method×fault×load combination
Boundary/Robustness Experiments	30 times per set

As shown in Table 1, the preliminary and formal experiments were completely separated. The core sample size was pre-determined using a two-sided two-sample test for independent proportions: with conservative effects of approximately 0.65 and 0.85 for B1 and TA-AHR respectively, the sample size was set $\alpha=0.05$ at $1-\beta=0.80$ approximately 70.85 runs per group, with an upward adjustment to 72 runs. SLO, threshold, window, fault parameters, and analysis scripts were frozen before the formal tests. Runs that experienced abnormal interruptions or missing critical logs were

marked as invalid and the reasons were recorded. The experimental order was randomized, and run_id was associated with method, fault, load, seed, commit, request, endpoint, transaction, and Chaos events, allowing charts and statistics to be reconstructed from data at the same run level.

2.2 Transaction-Aware Gray Fault Detection

TA-AHR refreshes the instance status every 1 second. Since a single Payment replica only receives approximately 5-6 payment calls per second under high load, this study does not directly calculate p99 within a 1-second sample, but instead calculates the rolling 99th quantile for call latency over the most recent 30 seconds; if the effective sample size is less than 50, this anomalous evidence is not updated. Error rate and timeout rate are calculated using a window of the most recent 10 seconds, and transaction progress age is derived from currently en route transactions. For a transaction i , its progress age is defined as :

$$TPA_i(t) = t - t_i^{\text{last}},$$

Where t_i^{last} represents the time of the most recent valid state change. The instance-level observation vector is :

$$X_t = [ER_t, TO_t, P99_t, TPA_t].$$

Preliminary, fault-free experimental data is used to calculate the 99th percentile thresholds for each indicator $q_{0.99}^j$. The definition of an indicator exceeding the normal range is as follows :

$$e_j(t) = I(x_j(t) > q_{0.99}^j),$$

The current number of anomalous evidence is :

$$E_t = \sum_{j=1}^4 e_j(t).$$

No manually assigned linear weights are used. A sampling point is classified as anomalous when $E_t \geq 2$. If at least six of the ten most recent sampling points are anomalous, the Payment replica transitions from HEALTHY to DEGRADED:

$$\sum_{\tau=t-9}^t I(E_\tau \geq 2) \geq 6.$$

Once an isolated replica enters RECOVERING, it receives only 1 Hz non-committing probe RPCs to assess network and service recovery; these probes are excluded from the calculation of business-service success rates. The replica transitions back to HEALTHY only if E_t remains at or below 1 for 15 consecutive seconds and the probe-latency p95 does not exceed 1.2 times its fault-free baseline.

2.3 Instance isolation, retry budget, and transaction-safe recovery

Once TA-AHR confirms that replica P_i has degraded, the Checkout-side instance routing table immediately stops allocating new payments to it:

$$\text{Route}(\tau_{\text{new}}) = P_j, \text{State}(P_j) = \text{HEALTHY}.$$

Unlike the static rule B1 that "each failed request will be retried a maximum of 2 times," TA-

AHR sets both a transaction-level cap and a system-level retry budget. Let ' $W_r=10s$ ' be the retry amplification factor:

$$RAF_t = \frac{N_{\text{retry}}(t-W_r, t)}{N_{\text{original}}(t-W_r, t)},$$

and constraints:

$$RAF_t \leq 0.25.$$

When the budget is exhausted or the HEALTHY Payment is not available in the system, the controller will no longer add recovery requests. This constraint is intended to break the amplification chain of "timeout $\rightarrow$ retries $\rightarrow$ extra load $\rightarrow$ more timeouts".

For transactions in abnormal instances that have not yet obtained a definite result, first check the transaction status, and the recovery rule is :

$$\text{Action}(\tau_i) = \begin{cases} \text{ReturnPrevious,} & s_i = \text{SUCCEEDED,} \\ \text{Reconcile,} & s_i = \text{PROCESSING,} \\ \text{Retry,} & s_i = \text{FAILED} \wedge \text{RetryAllowed,} \\ \text{FailClosed,} & \text{otherwise.} \end{cases}$$

When the same idempotent key is already SUCCEEDED, the system directly returns the original result. Transactions in the PROCESSING state that cannot be confirmed within 5 seconds are placed in a deferred queue. After the state storage is restored, it is checked again every 5 seconds, up to a maximum of 60 seconds. Transactions still unknown are transferred to asynchronous or manual verification. This study records the 5-second and 60-second states to distinguish between immediate availability loss and final disposal.

2.4 Control Methods and Experimental Protocol

Request-level failures need to limit the injection target and scope of impact [6]. F2 adds 1000 ms latency and about 100 ms jitter to a single Payment; F3 injects 30% related packet loss into a single replica, alternating between 5 s abnormal/5 s normal, which constitute the core experiment. F0/F1 are used for normal overhead and Pod Kill verification; F4 adds 1000 ms latency to both Payments simultaneously; F5 suppresses the response with a 10% probability after SUCCEEDED persistence; F6 uses NetworkChaos to block the Checkout/Payment to PostgreSQL TCP 5432 connection for 10 s; R1 linearly increases the single replica latency from 0 to 1000 ms within 30 s. Other scenarios only serve as security, boundary or robustness verification. See Table 2.

Table 2. Control methods and experimental scenarios/roles

category	set up	Research role
B0	Kubernetes Ready state + dual replicas	Infrastructure-level HA baseline
B1	B0+1200 ms timeout+2 times Retry+Circuit Breaker	Static toughness baseline
TA-AHR	B1 Basic Protection + Transaction Awareness Detection + Endpoint Isolation + Global Retry Budget + Transaction Verification	Proposed
F2/F3	single-replica persistent latency/intermittent packet loss	Core gray fault experiment
F0/F1	Normal / Pod Kill	Overhead and Hard Fault Verification

category	set up	Research role
F4/F6	No healthy copy/State storage unavailable	Capability Boundaries
F5	Response lost after submission	Restore security
R1	Progressive delay	No robustness of the model observed

B1 employs a 1200 ms timeout, a 10 s circuit breaker window, minimum calls=10, failure rate=50%, and OPEN for 5 s. Retry only applies to the timeout and 5xx parameters; common call parameters are frozen by an independent pre-experiment. Single- and dual-replica capacity scans yielded $Q_{sat}^{(1)} \approx 11.7$ and $Q_{sat}^{(2)} \approx 14.6$ transaction/s, respectively.. Medium and high loads were calculated as 0.50 and 0.75 times the dual-replica capacity, respectively, resulting in approximately 7.3 and 10.9 transactions/s. High load was approximately 93% of the single-replica saturation throughput, making the remaining capacity after isolation a real constraint.

The 800 ms SLO in this study is not an industry-standard threshold but a specified before the formal experiments experimental upper bound. The normal p99 upper limit under high load in the pre-experiment is approximately 260 ms. Therefore, rounding up to approximately three times the normal tail delay, 800 ms is set as the primary SLO, and 600 and 1000 ms are used for sensitivity calculations. The main business metrics are defined as follows :

$$A_{SLO} = \frac{N_{\text{success} \wedge \text{latency} \leq \text{SLO}}}{N_{\text{total}}} \times 100\%.$$

The detection success rate is :

$$DSR = \frac{N_{\text{detected}}}{N_{\text{runs}}} \times 100\%,$$

Detection delay is defined as $T_D = t_{\text{detect}} - t_{\text{fault}}$. Let t_s denote the start of the first 15-second stable interval during which $A_{SLO} \geq 99\%$ and p95 remains no greater than 1.2 times its fault-free baseline. Total fault exposure time is then defined as:

$$T_E = t_s - t_{\text{fault}}.$$

The statistical unit is independent operation. DSR reports the Wilson 95% CI and uses Fisher's exact test; undetected runs are right-censored after 30 seconds. T_D The median [IQR] of the condition is reported only if the number of successful detections is not less than 10; otherwise, it is recorded as NE. For continuous variables, the median [IQR] is reported. Core business metrics provide the 95% CI of the median difference of 5000 Bootstrap tests, and are tested using the Mann-Whitney U test. DSR and A_{SLO} Holm correction are applied in both F2 and F3 scenarios. Transaction safety is defined as :

$$DCTR = \frac{N_{\text{duplicatecommit}}}{N_{\text{ambiguous}}},$$

$$DRR_t = \frac{N_{\text{unresolvedat } t}}{N_{\text{ambiguous}}},$$

$$SRR_t = \frac{N_{\text{safelyresolvedby } t}}{N_{\text{ambiguous}}}.$$

This represents the number of transactions that $N_{\text{ambiguous}}$ denoted and have been successfully

persisted but for which Checkout has not received a definite response.

3. Research Results

3.1 Normal operating overhead and hard fault detection

Under high load on F0, the p99 values for B0, B1, and TA-AHR are 231.6 [224.1, 238.4], 248.3 [240.8, 257.9], and 251.7 [247.5, 260.5] ms, respectively. Compared to B0, TA-AHR increases by 8.7%, with a Bootstrap 95% CI of 5.8%–12.2%; compared to B1, the additional p99 is only 1.4%, with a 95% CI of -1.8%–3.8%. Stable throughput decreased from 10.89 [10.71, 11.07] transactions/s in B0 to 10.43 [10.24, 10.52] transactions/s in TA-AHR, CPU utilization increased from 46.0% [44.2%, 46.9%] to 48.6% [47.0%, 49.8%], and memory usage increased from 609.5 [598.1, 625.1] MB to 635.0 [628.8, 645.3] MB. Only one unnecessary isolation occurred in 30 normal operations, with a false isolation rate of 3.3% (Wilson 95% CI 0.6%–16.7%). These results show that business-level detection incurs measurable overhead, but TA-AHR does not have a significant additional tail latency burden compared to the statically resilient B1.

In F1, all three methods successfully recovered Pods from failures, with a median total failure exposure time of approximately 8 seconds, and the differences between groups were not statistically significant. This scenario is only used for hard failure verification, indicating that the incremental value of TA-AHR does not lie in fail-stop failures that Kubernetes can already directly identify.

3.2 Core Gray Fault Results

Table 3 presents the high-load results for the F2 and F3 scenarios. In F2, TA-AHR detected the fault in 61 of 72 runs, compared with 47 of 72 runs for B1, corresponding to DSRs of 84.7% and 65.3%, respectively. The difference was 19.4 percentage points and was statistically significant after Holm correction of the Fisher exact-test result ($p=0.012$). In F3, TA-AHR detected the fault in 54 of 72 runs, compared with 36 of 72 runs for B1, corresponding to DSRs of 75.0% and 50.0%, respectively. The difference was 25.0 percentage points and was also statistically significant after Holm correction ($p=0.007$). These results support a higher detection success rate for TA-AHR in both core gray-fault scenarios.

Table 3 Experimental Results of High-Load Core Gray Faults

Scene	method	DSR/% (Wilson 95% CI)	T_D /s median[IQR]	T_E /s median[IQR]	A_{SLO} /s median[IQR]
F2	B0	19.4 [12.0, 30.0]	26.5[26.0,26.9]	35.8[33.6,38.5]	84.6[80.1,87.5]
	B1	65.3[53.8,75.2]	17.2[15.7,19.1]	24.6[21.7,27.5]	89.6[86.5,93.5]
	TA-AHR	84.7[74.7,91.2]	13.6[11.5,16.1]	19.3[16.2,21.8]	91.3[88.5,94.4]
F3	B0	9.7[4.8,18.7]	NE	37.4[34.8,40.5]	77.8[74.3,83.7]
	B1	50.0[38.7,61.3]	21.0[19.3,22.9]	29.8[26.1,32.9]	83.3[77.7,87.2]
	TA-AHR	75.0[63.9,83.6]	15.0 [12.9, 19.0]	24.0[20.1,27.2]	87.1[80.1,90.6]

Note: T_D is summarized only for runs in which the fault was successfully detected. Groups with fewer than 10 successful detections are reported as NE (not estimable). B0 successfully detected the fault in 7 of 72 F3 runs.

In F2, TA-AHR's median A_{SLO} improved by 1.7 percentage points relative to B1, with a median difference Bootstrap 95% CI of 0.24–4.17 percentage points; in F3, it improved by 3.8 percentage points, with a 95% CI of 0.15–6.64 percentage points. The original p values of the two Mann-Whitney U comparisons in each scenario were 0.015 and 0.014, respectively, and were both corrected for Holm $p=0.028$. The improvement in detection did not proportionally translate into an increase in business availability. In F2, median detection delay T_D decreased from 17.2 to 13.6 s, while median total fault exposure time T_E decreased from 24.6 to 19.3 s. Results were more variable in F3, where TA-AHR failed to detect the fault within the 30-second fault window in 18 of 72 runs.

The effect weakens under medium load: Under medium load, the observed differences were smaller. In F2, DSR was 68.1% for B1 and 81.9% for TA-AHR, while median A_{SLO} was 94.2% and 95.3%, respectively. In F3, DSR was 52.8% for B1 and 69.4% for TA-AHR, while median A_{SLO} was 89.9% and 92.3%, respectively. Neither medium-load DSR comparison reached the 0.05 significance threshold after Holm correction. Therefore, the medium-load results indicate differences in the same direction as the high-load results, but provide insufficient statistical evidence to confirm an improvement.

Figure 1 illustrates the typical time variations for different mechanisms in F3. Representative runs were selected as those whose T_E was closest to the group median, and statistical judgment was based on the 72 replicates in Table 3.

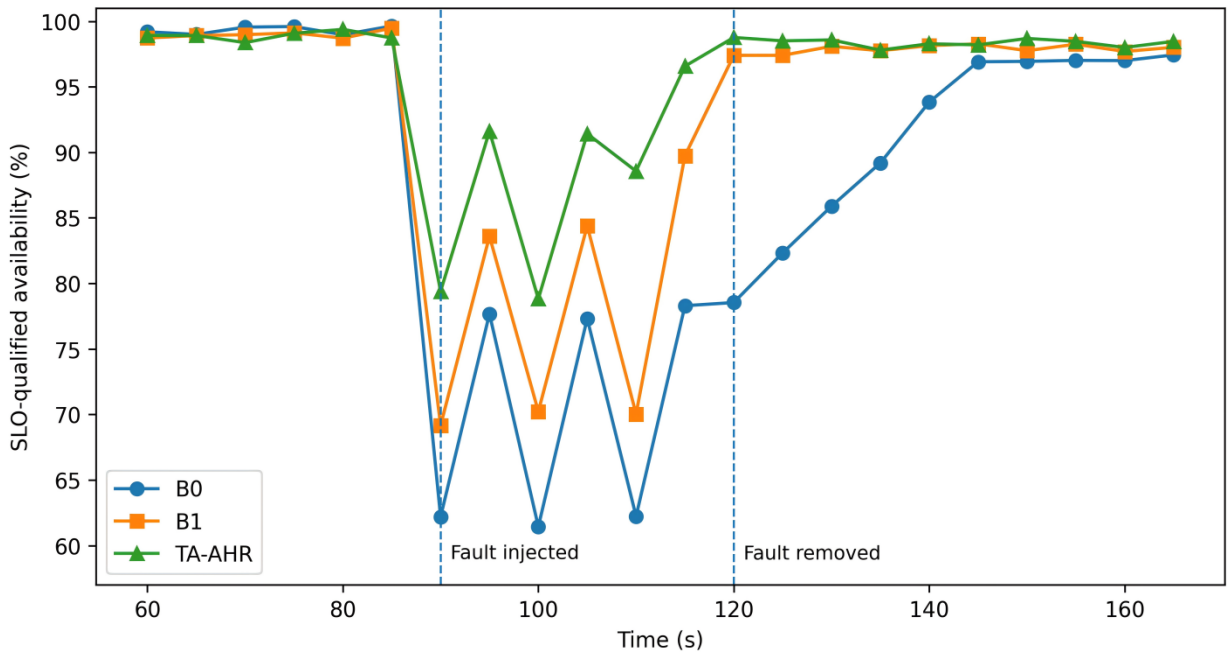


Figure 1. Changes in SLO availability during typical F3 experimental runs.

3.3 Restoring security, capability boundaries, and robustness

F5 constructs 400 indeterminate events for each scenario: "Payment persisted SUCCEEDED, but Checkout received no response." B1 produced a second valid commit in 347 of the 400 ambiguous transactions, corresponding to $DCTR=347/400=86.75\%$. No second valid commit was observed in the 400 TA-AHR runs. However, nine transactions remained unresolved after 5 s, corresponding to $DRR_{5s}=9/400=2.25\%$ and $SRR_{5s}=97.75\%$. Eight of these nine deferred transactions were safely resolved within 60 s, resulting in $SRR_{60s}=399/400=99.75\%$. The remaining transaction required sub-

sequent asynchronous or manual verification. After disabling transaction verification, DCTR rose back to approximately 84%, indicating that the safety result mainly stemmed from transaction state constraints, rather than gray fault detection itself.

In F6, state storage is inaccessible for 10 seconds. No second valid commit was observed in 200 uncertain transactions, but 31 entered the deferred queue $DRR_{5s}=15.5\%$; 27 were verified within 60 seconds after recovery, and 4 remain pending. The deferred queue is rechecked periodically according to Section 2.3 without blind retries; unrecoverable data loss exceeds the limits of the F6 model, and this study does not guarantee correctness. When two payments simultaneously degrade in F4, stable SLOs were not restored within the fault window, indicating that detection cannot replace healthy capacity.

Parameter robustness serves only as supplementary evidence. Under high-load F3 conditions, the DSRs for 8, 10, and 12 s windows were 83.3%, 76.7%, and 66.7%, respectively, with false-isolation rates under normal conditions of 13.3%, 6.7%, and 3.3%. At thresholds of $q_{0.975}$, $q_{0.99}$, and $q_{0.995}$, the DSRs were 80.0%, 76.7%, and 70.0%, respectively, with false-isolation rates of 10.0%, 6.7%, and 3.3%, showing a consistent trade-off between detection rate and false isolation. In the R1 progressive-latency scenario, the DSRs for TA-AHR and B1 were 70.0% and 46.7%, respectively, with Fisher's exact-test $p=0.115$; the corresponding A_{SLO} values were approximately 89.6% and 87.0%. These results were directionally consistent with the main experiment, but the evidence was insufficient to support the effectiveness of TA-AHR for arbitrary unknown gray-fault patterns.

When the business SLO threshold was set to 600, 800, and 1000 ms, respectively, the TA-AHR–B1 difference in A_{SLO} was approximately 2.4, 1.7, and 1.1 percentage points in F2, and 4.6, 3.8, and 2.3 percentage points in F3. The direction of the observed difference did not reverse across the three thresholds. Both groups had lower absolute availability under the stricter threshold, whereas the between-group difference narrowed under the more lenient threshold. Thus, the direction of the main result was not dependent on the single 800 ms threshold, although the magnitude of the observed availability improvement was sensitive to the SLO definition.

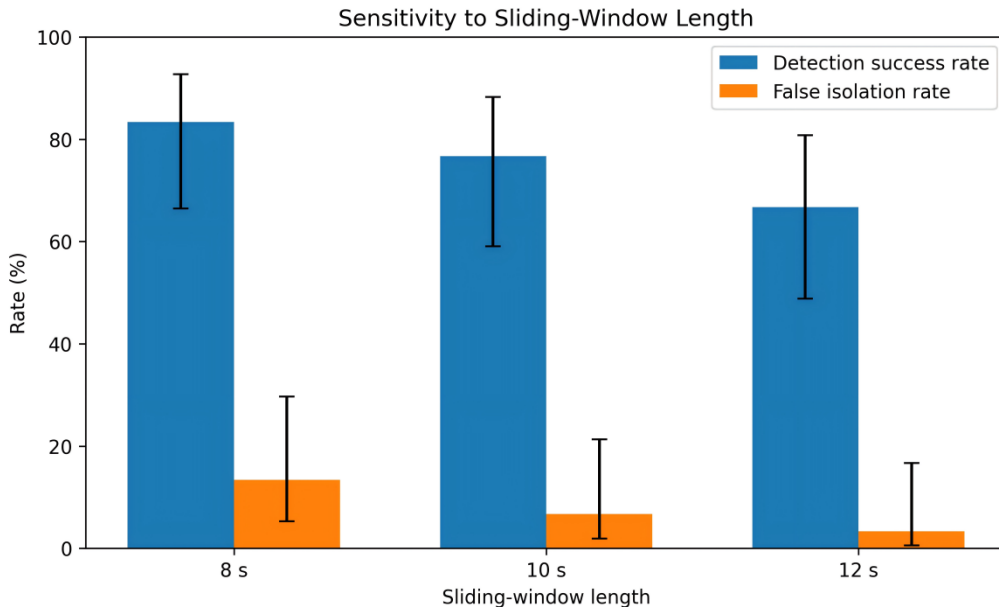


Figure 2. Detection success rate and false isolation rate under different sliding windows (Wilson 95% CI)

4. Discussion

The main experiment shows that the incremental value of TA-AHR mainly appears in gray failures where "the instance is still Ready, but the transaction quality has degraded", rather than replacing the existing fail-stop recovery in Kubernetes. Existing microservice anomaly research shows that runtime operation information can supplement the description of anomaly services by the state of a single infrastructure [7]; large-scale microservice performance diagnosis also emphasizes the importance of call dependencies and runtime context for performance anomaly identification [8]. This study further restricts this information to the runtime decision at the Payment endpoint level, so that the detection results directly drive instance isolation, rather than extending to general root cause diagnosis.

The improvement in DSR in F2 and F3 is significantly greater than the final A_{SLO} improvement. The reason is that there is already a service exposure time before the fault is confirmed, and after the abnormal instance is isolated, the healthy replica still has to take over the transferred traffic. Therefore, earlier identification is only a necessary condition for improving availability, not a sufficient condition. Multi-source telemetry studies have shown that different operational data can provide complementary fault information [9]; the F4 results of this study further illustrate that the observation capability is still constrained by the actual service capacity.

Payment security comes with a clear availability cost. In F5, B1 produced second valid commits in 86.75% of ambiguous transactions, whereas no second valid commits were observed under TA-AHR, with a 5-s deferred-resolution rate of 2.25%; however, this result applies only to response loss after SUCCEEDED had been reliably persisted. In F6, the delay rate continues to rise after the state store becomes temporarily unavailable, which shows that "no duplicate commits observed" cannot be evaluated separately from security dependencies and waiting costs. TA-AHR stops blindly retrying when the original transaction state cannot be confirmed. This processing is consistent with the engineering logic of limiting error propagation and invalid recovery behavior in microservice cascading failure research [10].

This study still has clear external validity boundaries. The experiments only cover gray failures of the Checkout-Payment single-replica model in the open prototype; adding replicas changes the capacity margin after failover, and R1 also shows no diminishing advantage in the time-based model. F6 only simulates temporary inaccessibility of the state store; unrecoverable data loss still requires independent persistent redundancy or external reconciliation, and TA-AHR does not assume database disaster recovery. Future research is more suitable for reusing the same protocol in a second payment platform and verifying the atomic persistence boundary, rather than continuing to expand the research topic.

5. Conclusion

This research narrows down the high availability problem of distributed payments to gray failures of a single-replica Payment, establishing a unified recovery relationship between transaction progress awareness, instance-level anomaly isolation, global retry budget, and unconfirmed transaction verification. Core experiments show that when the infrastructure still determines the replica is healthy, TA-AHR can improve the success rate of gray failure detection and improve the effective availability of SLOs, but its benefits are jointly limited by the remaining capacity of healthy replicas, normal operation overhead, and state storage availability. Transaction safety experiments further demonstrate that, under the condition that successful states have been reliably persisted, the recovery path can avoid blind retries that would form a second valid payment, but requires accepting a small number of transactions with deferred verification. The contribution of this research is not to cover all microservice failures, but to establish an achievable, measurable, reproducible, and clearly defined gray failure safe recovery closed loop for the critical payment chain.

References

- [1] NICOLAS-PLATA A, GONZALEZ-COMPEAN J L, SOSA-SOSA V J. A service mesh approach to integrate processing patterns into microservices applications[J]. *Cluster Computing*, 2024, 27(6): 7417–7438. DOI: 10.1007/s10586-024-04342-5.
- [2] AL-SAID AHMAD A, AL-QORA'N L F, ZAYED A. Exploring the impact of chaos engineering with various user loads on cloud native applications: An exploratory empirical study[J]. *Computing*, 2024, 106(7): 2389–2425. DOI: 10.1007/s00607-024-01292-z.
- [3] ADERALDO C M, COSTA T M, VASCONCELOS D M, MENDONÇA N C, CÁMARA J, GARLAN D. A declarative approach and benchmark tool for controlled evaluation of microservice resiliency patterns[J]. *Software: Practice and Experience*, 2025, 55(1): 170–192. DOI: 10.1002/spe.3368.
- [4] Wang, Z. (2026). Research on Operation Data Analysis and Efficiency Improvement of the New Energy Industry Based on Machine Learning. *Procedia Computer Science*, 285, 605-612.
- [5] Cui, N. (2024, December). Research on Adaptive Urban Traffic Green Wave Control Algorithm and Safety Based on Reinforcement Learning DQN-COP. In *International Conference on Big Data Analytics for Cyber-Physical System in Smart City* (pp. 379-387). Singapore: Springer Nature Singapore.
- [6] Xin, H. (2026). Research on Distributed Data Pipelines and Event-Driven Architecture for Freight Visibility.
- [7] Zhang, Z. (2026). User Behavior Device Reliability Joint Modeling and Intelligent Risk Control Practice In Consumer Technology.
- [8] Zhang, Z. (2026). Research on Service-oriented Packaging of AI Systems and Data-driven Decision Support for Real-world Business Processes. *Engineering Advances*, 6(3).
- [9] Wang, B. (2025, November). Research on Performance Improvement of Personalized Recommendation System for E-commerce Platform Based on Big Data and Collaborative Filtering Algorithm. In *International Conference on Computational Technologies for Research in Data Analytics* (pp. 36-46). Cham: Springer Nature Switzerland.
- [10] Qian, X. (2026, July). Application of the Combination Model of Prophet and LSTM in Multi-Level Inventory Demand Prediction for Retail Supply Chain of CPG-Consumer Packaged Foods. In *2026 5th International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)* (pp. 1-7). IEEE.
- [11] Zhang, Y. (2025, January). Research on the Application and Optimization of Multi Dimensional Data Model Based on Kylin in Enterprise Technology Management. In *International Conference on Innovative Computing* (pp. 234-241). Singapore: Springer Nature Singapore.
- [12] Fei, W. (2026). Families Of Chinese Background In Rural Alabama: Education, Immigration, And Identity.
- [13] Wendan Xue, Wen Fei. An Analytical Study on the Differences and Determinants of Basic Education Resource Allocation in Rural China and the United States. *International Journal of Educational Research and Development*, 1(3), 13-17.
- [14] Zhang, Z. (2026). User Behavior Anomaly Identification Integrating Telemetry Data and Its Application in Consumer Finance Risk Control.
- [15] Liu, S. (2026). Design and Implementation of Retrieval Enhancement Generation Method for Enterprise Knowledge Base.
- [16] Jin, C. (2026). Application of Data Mining Technology in Predicting User Churn and Retention Strategies in E-Commerce.
- [17] Jin, C. (2026). Construction and Empirical Study of Credit Risk Assessment Index System for Small and Micro Enterprises in Commercial Banks.

- [18] Fei, W., Xue, W., & Wang, B. (2025). *Research on the innovative path of rural education teaching methods from the perspective of PBE and PBL integration*. *Int. Acad. J. Humanit. Soc. Sci.*
- [19] Boli Wang, Wen Fei. *Innovations and practices in the classroom for rural aesthetic education in the perspective of school integration*. *International Journal of Educational Research and Development*, 3(1), 36-41.
- [20] Liu, S. (2026). *Automatic Generation Method and Implementation of Web Front-End Components Based on Generative AI*.
- [21] Jingzhi Yin. (2026). *Constructing a CMBS Default Risk Sentiment Index Using Financial Text Embeddings and Evaluating Its Predictive Effectiveness*. *Engineering Advances*, 6(1), 50-54.
- [22] Fei, W., & Qian, J. (2026). *Application and Exploration of Reverse Instructional Design in the Teaching of the Primary School Physical Education" Walking and Playing" Unit*. *Journal of Sociology and Education*, 2(2).
- [23] Zhang, Y. (2026). *Exploration of Enterprise Big Data Microservice Architecture Based on Domain-Driven Design (DDD)*. *Procedia Computer Science*, 282, 1994-2003.