

Modeling Loading Anomalies and Identifying Root Causes in Media Consumption Workflows on Large Social Media Platforms

Yuewei Yuan

RodericSchavone3768@outlook.com

University of Pennsylvania Philadelphia, United States 19104

Tianyi Xu

Georgetown University Washington, D.C., United States 20037

Jiayang Yin

Johns Hopkins University Baltimore, United States 21218

Jialei Huang

Columbia University New York, United States 10027

Research Article

Keywords: media consumption link, service dependency modeling, root cause identification, dynamic graph modeling, latency propagation, AIOps, platform reliability

Posted Date: September 3rd, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10900822/v1>

License:  This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: The authors declare no competing interests.

Modeling Loading Anomalies and Identifying Root Causes in Media Consumption Workflows on Large Social Media Platforms

Yuewei Yuan*

University of Pennsylvania
Philadelphia, United States 19104
Corresponding author: yuewei@alumni.upenn.edu

Jiayang Yin

Johns Hopkins University
Baltimore, United States 21218
jiyin33@jh.edu

Tianyi Xu

Georgetown University
Washington, D.C., United States 20037
tx52@georgetown.edu

Jialei Huang

Columbia University
New York, United States 10027
jh3882@columbia.edu

Abstract: Large-platform media workflows coordinate metadata, permission, cache, Uniform Resource Locator (URL), comment, interaction-status, and rendering services. Existing Root Cause Analysis (RCA) and Artificial Intelligence for IT Operations (AIOps) methods often rely on static topology. A dynamic service-dependency graph models active 5 min windows, ranks causes, and predicts 95th percentile (P95) delay. Eight weeks of data cover 91 services and 538 edges. Top-1/Top-3 accuracies are 0.872/0.941, and localization time falls from 27.6 to 6.4 min. The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are 86/143 ms. Comment blockage raises P95 latency to 2,380 ms; cache penetration lowers hit rate from 0.943 to 0.781 and produces a 99th percentile (P99) peak of 2,590 ms. Eleven of 17 release disturbances are identified one window before the client peak.

Keywords: media consumption link; service dependency modeling; root cause identification; dynamic graph modeling; latency propagation; AIOps; platform reliability

1 Introduction

Microservice diagnosis uses causal inference, change-point detection, tracing, and multimodal detection [1,2], but fixed thresholds and static topologies miss changing entry, cache, retry, and release states [3,4]. The proposed 5 min graph activates observed calls, combines server and client evidence, and predicts root causes and next-window P95 delay from eight weeks of data.

2 Mechanism of Loading Anomaly Propagation in Media Consumption Chains

Requests traverse entry, business, cache, storage, and rendering services. From a fixed universe of 91 nodes and 538 edges, each 5 min window activates only calls observed for the current media entry.

Node and edge states combine latency, errors, cache and release status, calls, retries, and error similarity. Persistent delay or a phased release can shift Application Programming Interface (API) response times before client failure, delay, or exit [5].

3 Methods for Modeling Dynamic Service Dependency Graphs and Identifying Root Causes

3.1 Construction of Time-Varying Service Dependency Graphs

The fixed global graph contains 91 service nodes and 538 candidate edges; each 5 min window extracts the active subgraph observed for the current request path.

Let the global service dependency graph be defined as:

$$G^* = (V^*, E^*) \quad (1)$$

where V^* denotes the fixed set of 91 service nodes and E^* denotes the fixed set of 538 candidate directed dependency edges identified during the complete observation period. For each 5 min window, the active graph is:

$$G_\tau = (V_\tau, E_\tau, X_\tau, Z_\tau, C_\tau, W_\tau) \quad (2)$$

where $V_\tau \subseteq V^*$ and $E_\tau \subseteq E^*$ are the active nodes and edges. X_τ , Z_τ , C_τ , and W_τ denote node, edge, client-feedback, and dependency-strength matrices; unobserved candidates remain inactive.

For each active edge $e_{ij} \in E_\tau$ from service node i to service node j , the dynamic dependency strength is calculated as:

$$w_{ij}^\tau = \frac{n_{ij}^\tau}{\sum_{k \in N_i^\tau} n_{ik}^\tau + \varepsilon} (1 + \rho_{ij}^\tau) (1 + r_{ij}^\tau) \phi_{ij}^\tau \quad (3)$$

where n_{ij}^τ is the call count from i to j ; N_i^τ is the active downstream-neighbor set of i ; and ε prevents division by zero. ρ_{ij}^τ is latency correlation and r_{ij}^τ is retry amplification.

The deterministic correction combines cache pressure, release disturbance, and error-code co-occurrence:

$$\phi_{ij}^\tau = 1 + (1/3)(c_{ij}^\tau + v_{ij}^\tau + e_{ij}^\tau) \quad (4)$$

where c_{ij}^τ , v_{ij}^τ , and e_{ij}^τ are normalized to $[0,1]$; equal weights retain an interpretable, parameter-free correction.

The cache-pressure term is calculated from the cache-miss ratio of the downstream node:

$$c_{ij}^\tau = \text{clip} \left(\frac{m_j^\tau - \tilde{m}_j}{m_j^{95} - \tilde{m}_j + \varepsilon}, 0, 1 \right) \quad (5)$$

where m_j^τ is the cache-miss ratio of node j in the current window, $\tilde{m}_j$ is the median cache-miss ratio estimated from the training windows, and m_j^{95} is the corresponding 95th percentile. References use prior eligible windows; values are clipped to $[0,1]$.

The version-release disturbance term is defined as:

$$v_{ij}^\tau = (u_i^\tau, u_j^\tau) \quad (6)$$

Release state spans deployment and recovery and clears after two normal windows.

Error-code co-occurrence is the endpoint histogram intersection:

$$e_{ij}^{\tau} = \sum_{q \in Q} \min(p_{i,q}^{\tau}, p_{j,q}^{\tau}) \quad (7)$$

where Q is the set of non-success error-code categories, and $p_{i,q}^{\tau}$ and $p_{j,q}^{\tau}$ are the proportions of error code q among all non-success responses of nodes i and j , respectively. The term approaches 1 as endpoint distributions converge.

3.2 Representation of Link Anomaly States and Propagation Modeling

Server and client signals define the anomaly state of entry link l in window τ as:

$$\Psi_l^{\tau} = \sigma \left(\mu_1 L_l^{\tau} + \mu_2 Q_l^{\tau} + \mu_3 B_l^{\tau} + \mu_4 K_l^{\tau} + \mu_5 P_l^{\tau} + \mu_6 U_l^{\tau} + \sum_{(a,b) \in B_l^{\tau}} \chi_{ab}^{\tau} T_{ab}^{\tau} \right) \quad (8)$$

Here, Ψ_l^{τ} is the composite anomaly state, $\sigma(\cdot)$ is the normalization mapping, L_l^{τ} is the node-delay offset, Q_l^{τ} is the error-code distribution shift, B_l^{τ} is the retry-spike intensity, K_l^{τ} is the cache-miss pressure, P_l^{τ} is the release-perturbation indicator, and U_l^{τ} is the feedback intensity derived from loading failures, rendering delays, and user exits. B_l^{τ} is the set of directed edges covered by the entry link, χ_{ab}^{τ} is the normalized dynamic dependency strength of edge (a,b) , and T_{ab}^{τ} is its delayed propagation intensity.

The coefficients $\mu_1, \dots, \mu_6$ are learned jointly and constrained to be nonnegative by:

$$\mu_j = \frac{\exp(\theta_j)}{\sum_{k=1}^6 \exp(\theta_k)}, \quad \sum_{j=1}^6 \mu_j = 1 \quad (9)$$

where all θ_j values are initialized to 0 and updated by backpropagation under the joint objective:

$$L = L_{rank} + \lambda_d L_{delay} + \lambda_2 \| \theta \|_2^2 \quad (10)$$

L_{rank} is the cross-entropy loss between the predicted and labeled root-cause nodes, and L_{delay} is the Huber loss between the predicted and observed P95 loading delays. The first term in Equation (10) is the cross-entropy loss between predicted and labeled root-cause nodes, and the second term is the Huber loss between predicted and observed P95 loading delays. The two objective weights are selected on the validation set from 0.1, 0.5, and 1.0 for the ranking term and 10^{-5} , 10^{-4} , and 10^{-3} for the delay term. All nine combinations are evaluated, and the combination with the lowest joint validation loss is retained. The learned coefficients remain fixed during testing. Link propagation therefore reflects the contribution of each anomaly source under the joint localization and delay-prediction objective rather than a manually weighted score. Cache penetration, permission-check timeout, and comment-preloading blockage can consequently produce different propagation patterns across media entry points [7,8].

3.3 Ranking Root Cause Candidates and Predicting Load Delay Risks

With the link anomaly state represented, root-cause identification is formulated as candidate-service ranking rather than single-point alerting. The model uses 148 million client loading anomaly events and 126 version-release records and scores candidates according to service-side anomaly severity, dependency-edge propagation, entry-scenario match, client-side interpretability, and next-window loading risk. For candidate node h in window τ , the ranking score is:

$$\Gamma_h^{\tau} = \text{softmax}_h \left(\alpha_1 \Delta_h^{\tau} + \alpha_2 \sum_{g \in P_h^{\tau}} \zeta_{gh}^{\tau} \kappa_{gh}^{\tau} + \alpha_3 e_m^{\tau} r_h^{\tau} + \alpha_4 \text{sim}(o_h^{\tau}, y_m^{\tau}) + \alpha_5 d_{m,95}^{\tau+1} \right) \quad (11)$$

Here, Δ_h^{τ} is the local anomaly intensity; P_h^{τ} is the set of upstream paths ending at node h ; ζ_{gh}^{τ} and κ_{gh}^{τ} are the propagation weight and anomaly-lag increment, respectively; e_m^{τ} and r_h^{τ} are the entry-scenario and service-role vectors; o_h^{τ} and y_m^{τ} are the service anomaly and client-feedback patterns; and $d_{m,95}^{\tau+1}$ is the predicted P95 loading delay for entry m in the next window. Labels require agreement between incident records and traces. Multi-root cases use normalized multi-hot targets; Top- k is correct if any accepted root appears among the first k candidates. Ambiguous cases are excluded.

Target-window data are excluded; fixed normalization and lagged cache references prevent leakage [9].

3.4 Online Diagnostic Process

Online diagnosis begins with continuous collection of media-consumption telemetry. As shown in Figure 1, distributed trace logs, cache-access logs, release records, and client-side event logs enter the stream-processing queue. The Window Aggregation Module merges link events, updates service-node states, and recalculates dependency-edge strength for each 5 min window. It then assigns one of four entry-scenario tags: media-detail page loading, video-playback page loading, comment expansion, or interaction-status refresh. The anomaly-detection module evaluates six states: metadata-read delay, cache penetration, permission-check timeout, comment-preloading blockage, interaction-status read failure, and release disturbance. A transient metric fluctuation is not treated as a root cause unless its node state, propagation path, and client feedback are jointly consistent. The AIOps alert output records the service name, entry type, propagation edge, error-code cluster, and remediation priority so that the engineering team can trace each alert along the active call path.

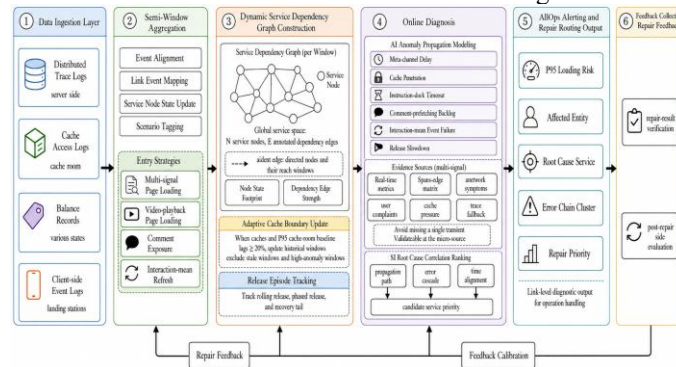


Figure 1 Flowchart of Online Diagnosis for Media Consumption Chains Driven by Dynamic Service Dependency Graphs

4 Experimental Results and Analysis

4.1 Dataset and Experimental Setup

Eight weeks of trace, cache, deployment, and client logs were aggregated into 5 min windows. Invalid identifiers, timestamps, mappings, and untraceable client events were removed. Each sample contains graph state, anomaly label, latency, errors, retry, cache and release state, rendering result, and exit time; labels were verified against tickets, remediation records, and traces.

The 8,064 windows were split chronologically into 5,040 training, 1,008 validation, and 2,016 test windows. Tickets, releases, and continuous anomaly episodes remained within one partition. Parameters used training data only, and cache references updated causally from earlier eligible windows.

The baselines were rule-based Median Absolute Deviation (Rule-MAD), Lagged Correlation (LagCorr), Static Personalized PageRank (Static-PPR), Long Short-Term Memory Autoencoder (LSTM-AE), Causal RCA, and Baro. All methods used identical metrics, validation selection, and evaluation measures.

4.2 Comparative Analysis of Root Cause Localization Performance

Time-series and static-graph baselines reached Top-1 accuracies of 0.804 and 0.756. The dynamic model achieved Top-1/Top-3 accuracies of 0.872/0.941 and 6.4 min localization. Figure 2 gives the 95% joint confidence region from 1,000 bootstrap samples, calculated as:

$$(z - \bar{z})^T \Sigma^{-1} (z - \bar{z}) \leq \chi_{2,0.95}^2 \quad (12)$$

where $z = [A, T]^T$, A is Top-1 Accuracy, T is Mean Localization Time, $\bar{z}$ is the bootstrap mean vector, Σ is the corresponding 2×2 covariance matrix, and $\chi_{2,0.95}^2 = 5.991$. The proposed method occupies the high-accuracy, low-latency region because active-edge weighting, entry-scenario

constraints, and client feedback suppress inactive or functionally unrelated candidates.

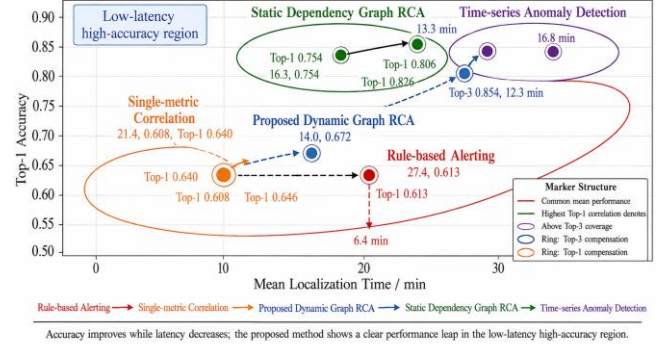


Figure 2 Comparison of two-dimensional confidence regions for root-cause localization accuracy and diagnostic latency

4.3 Analysis of P95 Load Latency Prediction Performance at the Link Level

Figure 3 evaluates whether the root-cause ranking model captures link-level P95 loading-delay changes in end-user experience. Prediction errors are concentrated within 1,200–2,400 ms, with most samples close to the 45° reference line. The proposed dynamic graph model achieved an MAE of 86 ms and an RMSE of 143 ms. In comparison, rule-based alerts reached 216 and 337 ms and failed to capture post-cache-penetration tail latency; single-metric correlation reached 184 and 291 ms but could not distinguish propagation across entry scenarios; and time-series anomaly detection reached 119 and 196 ms but captured only local spikes. By combining service-node states, dynamic dependencies, retry behavior, cache states, and client-rendering feedback, the proposed model reduced systematic underestimation of high-latency samples in video-playback and comment-expansion paths.

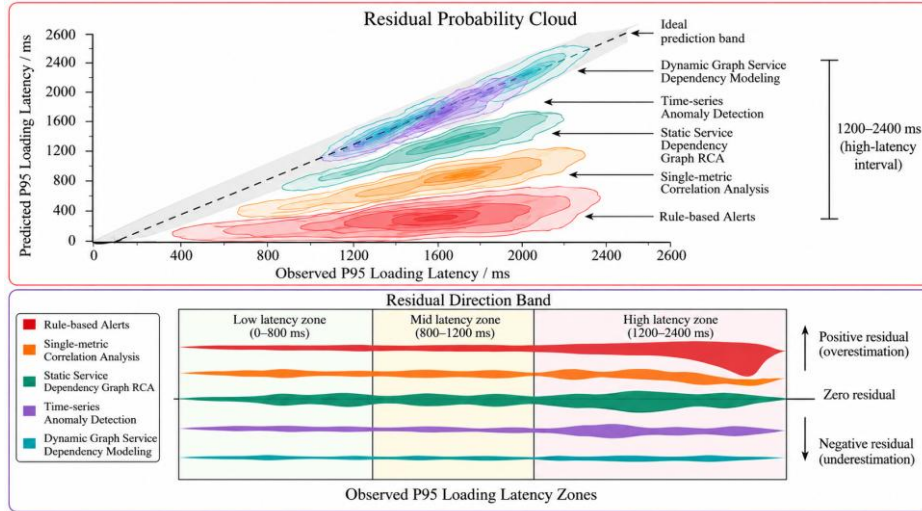


Figure 3 Density of P95 loading latency prediction errors at the link level—calibration residual plot

4.4 Analysis of the Propagation Effects of Different Anomaly Types

Figure 4 aligns anomalies at T0. Metadata delay, permission timeout, and interaction failure raise P95 latency to 1,960, 1,840, and 1,670 ms. Comment blockage raises it

from 980 to 2,380 ms and exit rate from 0.091 to 0.147. Cache hit rate falls from 0.943 to 0.781, with P99 latency of 2,590 ms at T+2. Eleven of 17 release disturbances are detected at T-1.

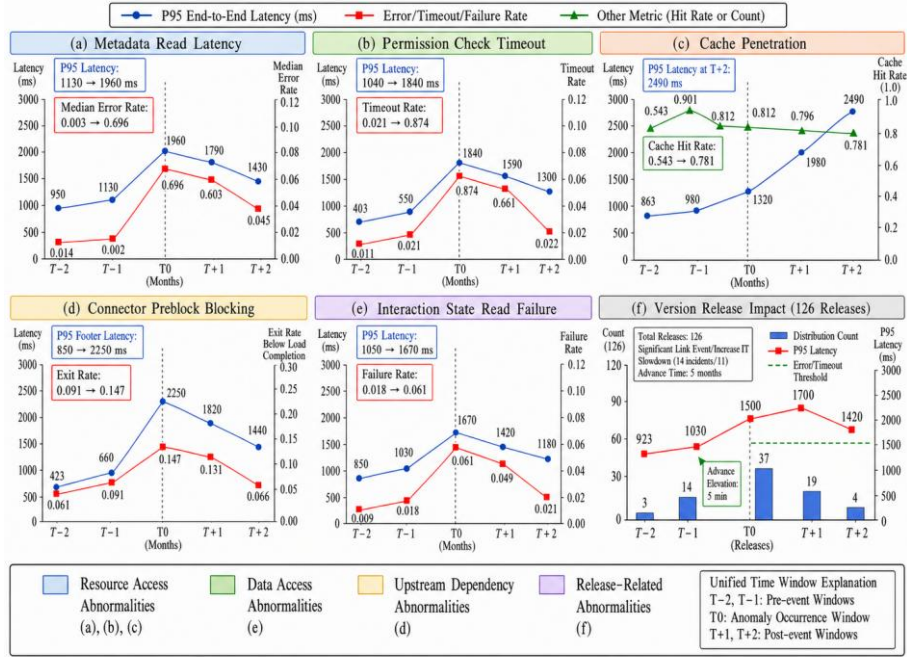


Figure 4 Event-aligned propagation fingerprints of loading anomalies from T-2 to T+2

4.5 Ablation Experiments and Stability Analysis

Table 1 shows the full model performs best. Removing dynamic edge weights lowers Top-1 accuracy from 0.872 to 0.803, raises localization time from 6.4 to 13.6 min, and

raises RMSE from 143 to 203 ms. Client-feedback removal increases localization time to 10.8 min. Dynamic edge strength is therefore the main mechanism separating active paths from nearby inactive dependencies.

Table 1 Ablation Experiment Results for the Dynamic Service Dependency Graph Model

Model variant	Removed component	Top-1 Accuracy	Top-3 Accuracy	Mean Localization Time (min)	MAE/ms	RMSE/ms
Full model	None	0.872	0.941	6.4	86	143
Without Client Feedback	Client-side loading feedback	0.821	0.902	10.8	112	181
Without Dynamic Edge Weight	Time-varying dependency strength	0.803	0.887	13.6	128	203
Without Cache State	Cache hit and penetration state	0.826	0.906	11.5	119	194
Without Retry Feature	Retry amplification feature	0.839	0.915	9.7	106	176
Without Release State	Version release status flag	0.847	0.921	8.9	101	169
Without Entry Scenario	Media entry encoding	0.814	0.895	12.2	124	198

5 Conclusion

The time-varying graph integrates cache, retry, release, and client evidence within the active media path. It achieves Top-1/Top-3 accuracies of 0.872/0.941, 6.4 min localization, and P95 errors of 86 ms MAE and 143 ms RMSE. Adaptive references represent drift without target-window labels. Evaluation remains limited to one platform and ticket-based labels; cross-platform traces and online graph updates require further validation.

References:

- [1] Xin R, Chen P, Zhao Z. Causalrca: Causal inference based precise fine-grained root cause localization for microservice applications[J]. Journal of Systems and Software, 2023, 203: 111724.
- [2] Zhang S, Xia S, Fan W, et al. Failure diagnosis in microservice systems: A comprehensive survey and analysis[J]. ACM Transactions on Software Engineering and Methodology, 2025, 35(1): 1-55.
- [3] Behera A, Panigrahi C R, Behera S, et al. trACE-Anomaly Correlation Engine for Tracing the Root Cause on Cloud Based Microservice Architecture[J]. Computación y Sistemas, 2023, 27(3): 791-800.
- [4] Nobre J, Pires E J S, Reis A. Anomaly detection in microservice-based

- systems[J]. Applied Sciences, 2023, 13(13): 7891.
- [5] Kabamba H M, Khouzam M, Dagenais M R. Vnode: Low-Overhead Transparent Tracing of Node. js-Based Microservice Architectures[J]. Future Internet, 2024, 16(1): 13.
- [6] Mazraemolla Z P, Rasoolzadegan A. An effective failure detection method for microservice-based systems using distributed tracing data[J]. Engineering Applications of Artificial Intelligence, 2024, 133: 108558.
- [7] Wei X, Sun C, Yang P, et al. TraLogAnomaly: A microservice system anomaly detection approach based on hybrid event sequences[J]. Science of Computer Programming, 2025, 245: 103303.
- [8] Fan M, Zhang X, Wang P, et al. Multi-modal anomaly detection for microservice system through nested graph diffusion reconstruction[J]. Applied Intelligence, 2025, 55(11): 784.
- [9] Pham L, Ha H, Zhang H. Baro: Robust root cause analysis for microservices via multivariate bayesian online change point detection[J]. Proceedings of the ACM on Software Engineering, 2024, 1(FSE): 2214-2237.
- [10] De la Cruz Cabello M, Sales T P, Machado M R. AIOps for log anomaly detection in the era of LLMs: A systematic literature review[J]. Intelligent Systems with Applications, 2025: 200608.