Journal of Computer Science and Electrical Engineering
Print ISSN: 2663-1938
Online ISSN: 2663-1946
DOI: https://doi.org/10.61784/jcsee4157

DEPENDENCY-AWARE ROOT CAUSE ANALYSIS AND AUTOMATED FAULT RECOVERY FOR CLOUD-NATIVE MICROSERVICES

Lin Zhou[1*], Xiang He[2]
[1]*University of Florida, Gainesville 32611, FL, USA.*
[2]*University of Illinois Urbana-Champaign, Urbana 61801, IL, USA.*
[*]*Corresponding Author: Lin Zhou*

Abstract: Failures in cloud-native microservice systems propagate along service dependencies, so the services that raise alarms are frequently victims rather than culprits, and a diagnosis alone does not restore service. This paper presents DARE, a dependency-aware framework that unifies root cause analysis (RCA) and automated recovery. DARE converts heterogeneous metrics, distributed traces, and logs into history-calibrated evidence that deflates routinely noisy channels; it separates intrinsic from inherited anomalies through exclusive span latency, error-origin attribution, propagation-aware residuals, and latent-state inference for uninstrumented services; and it ranks root-cause hypotheses with a symptom-gated likelihood-ratio model over a two-layer (call and co-location) dependency graph, which also yields a fault-type belief. A decision-theoretic planner, DARE-Heal, turns this belief into recovery actions by maximizing the success probability per unit cost, where the cost includes the blast radius of each action, and refines its belief after every unsuccessful attempt. On the public GAIA MicroSS dataset (382 labeled incidents, 28.7 million spans), the RCD Sock-Shop dataset, and a multimodal benchmark of 200 cascading-failure incidents, DARE attains top-1 accuracies of 0.817, 1.000, and 0.930 and matches or exceeds eight published baselines, including CIRCA and RCD, on every dataset. In closed-loop experiments, DARE-Heal recovers 86.7% of the GAIA incidents within five actions and reduces the mean time to recovery by 29.1% (GAIA) and 78.9% (synthetic) relative to restart-based remediation.
Keywords: Microservices; Root cause analysis; Distributed tracing; AIOps; Self-healing; Automated fault recovery

1 INTRODUCTION

Cloud-native applications are composed of tens to thousands of loosely coupled microservices that are deployed in containers, scaled independently, and connected by remote procedure calls [1-2]. This architecture accelerates delivery, but it also turns every incident into a distributed problem: a fault injected by a resource hog, a misconfiguration, or a defective release in one service propagates through synchronous calls, retries, and shared hosts, and it surfaces as latency and error alarms at many other services [3-4]. Operators then face two coupled questions under time pressure: *which service is the root cause*, and *which action restores the service fastest with the least collateral damage*.

A large body of work localizes root causes from monitoring data. Metric-based methods learn causal graphs with the PC algorithm and rank services by random walks, personalized PageRank, intervention recognition, or regression-based hypothesis testing [5-11]. Trace-based methods exploit end-to-end request paths to contrast abnormal and normal traces or to follow anomaly propagation chains [12-15]. Four obstacles limit these methods in production telemetry. (i) *Heterogeneous noise*: the deviation statistics of different channels are not comparable; network counters and tail latencies deviate routinely, so magnitude-based rankings are dominated by noisy channels. (ii) *Inherited symptoms*: callers inherit latency and errors from their callees and may even saturate their own resources through back-pressure, so symptoms resemble causes [4]. (iii) *Partial observability*: tracing and logging coverage is incomplete, and the faulty service may be uninstrumented. (iv) *The diagnosis–action gap*: a ranked list is not a remediation plan; restarting the top-ranked services wastes actions when the fault type calls for scaling, migration, or rollback, and every action on a heavily used service disrupts its callers.

This paper presents DARE (Dependency-Aware Root-cause analysis and rEcovery), an unsupervised framework that addresses these obstacles jointly (Figure 1). DARE first learns, from fault-free history only, how strongly each telemetry channel of each service deviates under normal operation, and deflates the evidence of routinely noisy channels. It then attributes anomalies to the services that generate them, using exclusive span latency and error-origin spans derived from distributed traces, propagation-aware residuals for services without span-level attribution, and an explicit latent state for uninstrumented services. Root-cause hypotheses are scored with a likelihood ratio under a *typed* and *symptom-gated* propagation model over a two-layer dependency graph: call edges carry latency and errors toward callers and traffic shifts toward callees, co-location edges carry resource contention, and every propagation path is active only to the extent that the hypothesized root itself exhibits the corresponding outward-visible degradation. Finally, DARE-Heal, a decision-theoretic planner in the tradition of Heckerman et al., converts the posterior over (root, fault type) into recovery actions, weighs each action by its success probability per unit cost including its blast radius on the dependency graph, and updates its belief by Bayes' rule after every verification window [16].

© By the Author(s) 2022, under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0).

The main contributions are as follows.
• A history-calibrated multimodal evidence model that makes metric, trace, and log deviations comparable without fault labels.
• A dependency-aware intrinsic attribution scheme and a symptom-gated, typed propagation likelihood over call and co-location dependencies, including latent-state inference for uninstrumented services.
• DARE-Heal, a closed-loop recovery planner that couples the diagnosis posterior with runbook knowledge, blast-radius-aware action costs, and Bayesian belief refinement.
• An evaluation on two public datasets and a multimodal cascading-failure benchmark against eight published baselines. DARE matches or exceeds all baselines on every dataset (0.817 top-1 accuracy on GAIA versus 0.743 for the strongest baseline), and DARE-Heal reduces the mean time to recovery by 29.1%–78.9% relative to restart-based remediation.
The remainder of the paper is organized as follows. Section 2 reviews related work, Section 3 formalizes the problem, Sections 4 and 5 present the diagnosis and recovery components, Sections 6 and 7 report the experimental setup and results, Section 8 discusses deployment aspects, and Section 9 concludes.

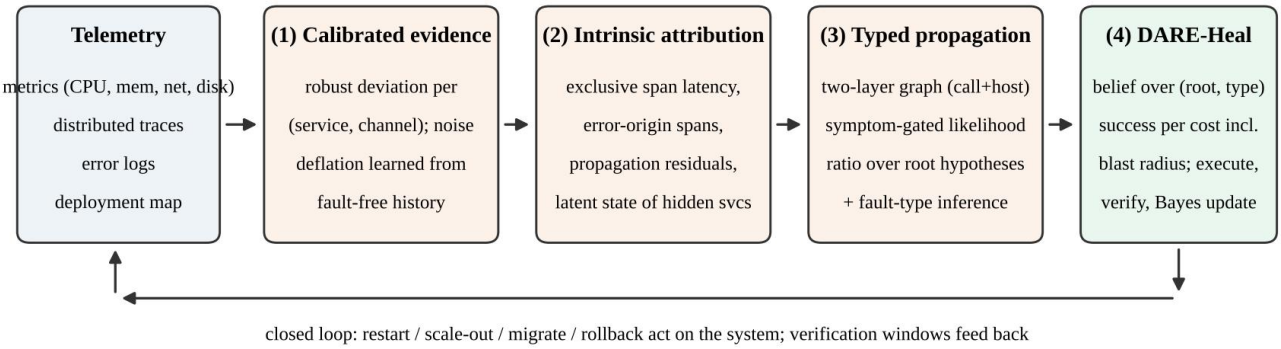


Figure 1 Overview of DARE

Note: Telemetry is converted into history-calibrated evidence, anomalies are attributed to the services that generate them, root-cause hypotheses are scored with typed and symptom-gated propagation over the call and host layers, and DARE-Heal executes and verifies recovery actions in a closed loop.

2 RELATED WORK

2.1 Metric-Based Root Cause Analysis

CloudRanger constructs an impact graph with the PC algorithm and applies a second-order random walk guided by correlations with the front-end KPI; MicroCause combines a path-condition time-series variant of PC with a temporal cause-oriented random walk; AutoMAP builds behavior graphs from multiple metric types [5-7, 11]. MicroRCA extracts an anomalous subgraph and ranks services by a personalized PageRank over an attributed graph of services and hosts [8]. ε-Diagnosis applies two-sample tests based on the ε-statistic to metrics of a small anomalous window [17]. RCD treats a failure as an intervention, adds an indicator node that separates normal from anomalous data, and learns only the neighborhood of this node with a localized and hierarchical PC procedure [9]. CIRCA formulates root cause analysis as intervention recognition and tests, for each metric, whether its distribution conditioned on its parents in a structural graph derived from the architecture has changed [10]. Seer and Sage apply deep learning and causal Bayesian networks with counterfactual queries to performance debugging in microservices [3-4]. DARE shares the structural view of CIRCA and Sage, but it scores *service-level* hypotheses with typed and gated propagation, calibrates evidence across modalities, and closes the loop with recovery [4, 10].

2.2 Trace-Based and Multimodal Localization

Distributed tracing records the causal path of every request [12]. TraceRCA ranks microservices that appear in many abnormal and few normal traces, MicroRank weights a spectrum analysis of normal and abnormal traces with PageRank, and MicroHECL analyzes anomaly propagation chains on a dynamically constructed call graph [13-15]. DejaVu learns, in a supervised manner, to localize recurring failures from multimodal data and failure dependency graphs [18]. DARE needs no labeled failures and uses traces for span-level attribution (exclusive latency and error origin) rather than for trace coverage statistics alone. Soldani and Brogi survey anomaly detection and root cause analysis for multi-service applications [1].

2.3 Dependency Inference and Automated Recovery

Sherlock models multi-level dependencies in enterprise networks with an inference graph and searches for the assignment of faulty components that best explains the observations; DARE adopts this explanatory view for microservices and adds symptom gating, typed propagation, and latent states [19]. Autonomic computing frames self-healing as a monitor–analyze–plan–execute loop [20]. Microrebooting shows that fine-grained restarts cure many

failures cheaply, and decision-theoretic troubleshooting proves that, under a single-fault assumption, repairing components in decreasing order of the ratio between failure probability and repair cost is optimal [16, 21]. DARE-Heal instantiates these principles with fault-type-specific actions, dependency-aware disruption costs, and a belief derived from multimodal telemetry.

3 PROBLEM FORMULATION

A microservice system consists of a set of services V deployed on hosts. The *call layer* is a directed graph $G_c = (V, E_c)$ with an edge (u, v) when u invokes v; it is recovered from parent–child relations of spans or from the reference architecture. The *host layer* maps every service to its host $h(v)$; services on the same host share CPU, memory, disk, and network capacity. Each service v exposes a set of telemetry channels C_v; every channel c has a kind $k(c)$ among CPU, memory, network, and disk utilization, exclusive span latency (excl), originating span errors (eorig), error logs (log), inclusive latency (lat), error ratio (err), and throughput (cnt). Channels may be missing because of partial instrumentation.

An incident is detected at time t_0. DARE compares an anomaly window $W_a = [t_0, t_0 + 10\ \text{min})$ with a baseline window $W_b = [t_0 - 60\ \text{min}, t_0 - 2\ \text{min})$. The tasks are (T1) to rank V by the likelihood of being the root cause r^*; (T2) to infer the fault type $f^* \in$ {CPU, memory, latency, disk, application error}; and (T3) to select a sequence of (service, action) pairs, with actions $a \in$ {restart, scale-out, migrate, rollback}, that minimizes the expected time to recovery and the disruption imposed on dependent services, within a budget of K actions after which the incident is escalated to an on-call engineer.

4 DARE: DEPENDENCY-AWARE DIAGNOSIS

4.1 History-Calibrated Multimodal Evidence

For each service v and channel c, DARE measures how far the anomaly window departs from the baseline with a robust standardized deviation

$$d_{v,c} = (\bar{x}_{v,c} - m_{v,c}) / (1.4826 \cdot \mathrm{MAD}_{v,c} + \alpha |m_{v,c}| + \varphi_{k(c)}), \tag{1}$$

where $\bar{x}_{v,c}$ is the mean over W_a, $m_{v,c}$ and $\mathrm{MAD}_{v,c}$ are the median and median absolute deviation over W_b, $\alpha = 0.05$ adds a relative floor that suppresses slow drift, and φ_k is an absolute floor per channel kind. For latency, error, and log channels only increases are harmful, so the one-sided deviation $\max(d, 0)$ is used; for the other kinds $|d|$ is used. Deviations of different channels are still not comparable: tail latencies and network counters exceed three robust standard deviations routinely, whereas memory utilization rarely does. DARE therefore estimates, for every (service, channel) pair, the 0.99-quantile $q_{v,c}$ of the deviation over a set of fault-free windows drawn from history, and deflates the evidence of channels that are noisy under normal operation:

$$s_{v,c} = |d_{v,c}| \cdot \tau / \max(q_{v,c}, \tau), \tag{2}$$

so that the normal 0.99-quantile of a noisy channel is mapped onto the decision threshold $\tau = 3$ while quiet channels remain unchanged. Calibration needs only fault-free history, not labels. Each score is converted into a soft anomaly indicator $o(s) = 1 - \exp(-\max(0, s - \tau)/\kappa)$ with $\kappa = 3$, and channels are pooled into three observation groups: service-level symptoms SYM = {lat, err}, traffic TRAF = {cnt, net}, and local resources RES = {cpu, mem, disk, excl, eorig, log}. The group indicator is

$$o_{v,g} = \max_{c \in C_\{v,\, k(c) \in g\}} o(s_{v,c}). \tag{3}$$

4.2 Dependency-Aware Intrinsic Attribution

Inclusive latency and error ratios of a caller contain the contributions of its callees. DARE therefore derives two *intrinsic* trace features per span by joining spans across services through their parent identifiers: the exclusive latency, i.e., the span duration minus the durations of its child spans, and the error origin, i.e., an erroneous span none of whose children is erroneous. Both are aggregated per service and minute. For services whose latency is observed only at the service level, DARE computes a propagation-aware residual

$$\rho_v = \max(0, s_v^{\mathrm{lat}} - \max_{u \in \mathrm{Ch}(v)} s_u^{\mathrm{lat}}), \tag{4}$$

where $\mathrm{Ch}(v)$ is the set of callees; the residual is used only when all callees are observable, because a hidden callee makes the attribution ambiguous. The intrinsic magnitude I_v is the largest calibrated score among the RES channels and the residual. Uninstrumented services have no SYM observations, yet their service-level state is reflected in their callers. DARE represents it explicitly as a latent indicator

$$\tilde{o}_{v,\mathrm{SYM}} = \rho_{\mathrm{up}} \cdot \max_{u \in \mathrm{Pa}(v)} o_{u,\mathrm{SYM}}, \tag{5}$$

where $\mathrm{Pa}(v)$ is the set of callers; the latent indicator is used only when v itself is the hypothesized root, so that a hidden service can explain the symptoms of its callers.

4.3 Typed, Symptom-Gated Propagation Scoring

For a hypothesis that r is the root cause, DARE predicts the probability that group g of service v is anomalous. Call edges carry service-level symptoms toward callers and traffic shifts toward callees; host edges carry resource contention; and callers may also saturate their resources through back-pressure [4]. Each path is *gated* by the state of r: callers

inherit symptoms only if r itself degrades at the service level, and co-located services suffer contention only if r shows resource anomalies. With gates $\gamma_g = o_{r,g}$ and background rates ε_g, the predictions are

$$\pi_r(v,\mathrm{SYM}) = \varepsilon + (\rho_{up}^{d\uparrow} - \varepsilon)\cdot\gamma_{SYM}, \quad v \in \mathrm{Anc}(r), \tag{6}$$

$$\pi_r(v,\mathrm{TRAF}) = \varepsilon + (\rho_{down}^{d\downarrow} - \varepsilon)\cdot\max(\gamma_{SYM}, \gamma_{TRAF}), \quad v \in \mathrm{Desc}(r), \tag{7}$$

$$\pi_r(v,\mathrm{RES}) = \varepsilon + (\eta - \varepsilon)\cdot\gamma_{RES} \text{ if } h(v) = h(r), \text{ else } \varepsilon + (\rho_{bp}^{d\uparrow} - \varepsilon)\cdot\gamma_{SYM}, \tag{8}$$

where $\mathrm{Anc}(r)$ and $\mathrm{Desc}(r)$ are the transitive callers and callees of r, $d\uparrow$ and $d\downarrow$ are the corresponding hop distances, and caller traffic follows $0.5\cdot\rho_{up}^{d\uparrow}$. The background rate ε_g is set adaptively to the median group indicator across services, clipped to [0.02, 0.3]. The score of r is a log-likelihood ratio against the background model plus the intrinsic magnitude:

$$S(r) = \Sigma_g\, o_{r,g} \ln((1-\delta)/\varepsilon_g) + (1 - \max_g o_{r,g}) \ln(\delta/(1-\varepsilon)) + \Sigma_{v\neq r} \Sigma_g\, o_{v,g} \ln(\pi_r(v,g)/\varepsilon_g) + \lambda \ln(1 + I_r). \tag{9}$$

The first two terms reward a hypothesis that explains every anomalous group of the root itself and penalize a root without any visible anomaly. The third term credits the anomalies of other services that the hypothesis accounts for; the absence of predicted propagation is deliberately not penalized, because attenuation through call fractions, caches, and timeouts is unknown. Gating encodes the physical precondition of propagation: a resource fault in a widely used back-end, such as a Redis access service, that does not slow down its responses leaves its callers unaffected, and a hypothesis receives credit for caller symptoms only when its own service-level behavior degrades. Defaults are $\rho_{up} = 0.7$, $\rho_{down} = 0.3$, $\rho_{bp} = 0.3$, $\eta = 0.3$, $\delta = 0.05$, and $\lambda = 1$ for all datasets.

4.4 Fault-Type Inference and Complexity

Generic signatures such as CPU and latency frequently appear as side effects: memory stressors also consume CPU, and retries raise the CPU utilization of callers. DARE therefore assigns the fault type by *specificity-first* matching at the candidate root: application error (error origin, error logs), memory, disk, network, CPU, and latency are tested in this order, and the first type whose soft evidence reaches 0.9 is selected; otherwise the type with the largest evidence is chosen. The resulting type belief places 0.6 of the mass on the selected type and distributes the rest in proportion to the soft evidence of the other types. Scoring requires one breadth-first search per hypothesis, i.e., $O(|V|(|V| + |E_c|))$ operations per incident, and evaluates in milliseconds for systems with hundreds of services.

5 DARE-HEAL: CLOSED-LOOP RECOVERY

DARE-Heal maintains a belief over root–type pairs, $b(s, f) \propto P(s)\cdot P(f \mid s)$, restricted to the five most likely roots, with $P(s) \propto \exp(S(s)/T)$. The temperature T is fitted on the synthetic development set only ($T = 1$). Runbook knowledge is encoded as the probability $E(f, a)$ that action a resolves a fault of type f (Table 1); operators can edit these priors. For a candidate pair (s, a) the success probability and the cost are

$$p(s,a) = \Sigma_f\, b(s,f)\cdot E(f,a), \qquad C(s,a) = t_a + t_{ver} + \mu\cdot D_a\cdot(1 + B(s)), \tag{10}$$

where t_a is the execution time of the action, $t_{ver} = 60$ s is the verification window, D_a is the downtime the action imposes on s, $\mu = 0.5$, and $B(s) = \Sigma_{u \in \mathrm{Anc}(s)}\, 0.7^{d(u,s)}$ is the blast radius, i.e., the attenuated number of services that depend on s. Following the optimality result for single-fault troubleshooting [16], DARE-Heal executes

$$(s^*, a^*) = \mathrm{argmax}_{s,a}\, p(s,a) / C(s,a). \tag{11}$$

After the verification window, DARE re-evaluates the evidence of the affected services. If the symptoms persist, the belief is updated by Bayes' rule, $b(s^*, f) \leftarrow b(s^*, f)\cdot(1 - E(f, a^*))$, followed by normalization; the next action is then selected on the refined belief. The loop stops when recovery is verified or when $K = 5$ actions have been executed, after which the incident is escalated. Unlike restart-only remediation, the planner chooses fault-specific actions, prefers low-disruption actions on services with many dependents, and naturally switches to an alternative action or candidate when an attempt fails.

Table 1 Runbook Prior E(f, a) and Action Parameters

Fault type	Restart	Scale-out	Migrate	Rollback
CPU	0.55	0.80	0.70	0.20
Memory	0.90	0.30	0.50	0.40
Latency	0.15	0.35	0.75	0.20
Disk	0.20	0.25	0.85	0.10
Application error	0.30	0.05	0.10	0.90
Duration t_a (s)	40	60	90	120
Downtime D_a (s)	30	0	30	20

6 EXPERIMENTAL SETUP

6.1 Datasets

Table 2 summarizes the datasets. *Sock-Shop* is the public release of RCD, collected in 2022 on the Sock Shop application running on Kubernetes: CPU and memory hogs are injected into five services with five runs each, and Prometheus provides per-second CPU and memory metrics of 13 containers plus latency percentiles of three services [9, 22]. Normal and anomalous periods were collected separately. The call graph follows the reference architecture.

GAIA is the MicroSS system of the GAIA dataset released by CloudWise: ten services (web, mobile, log, Redis-access, and database-access services with two replicas each) run on four hosts [23]. We processed the July 2021 release, which provides container metrics, 28.7 million spans, and a log of injected faults. Faults include memory hogs, CPU hogs, file-missing errors in the web services, and access-denied errors in the database services; QR-code login failures are present as continuous background warnings and are kept as realistic noise. We use every injection without another injection within ±10 minutes, which yields 382 single-root incidents (345 memory, 19 file missing, 11 access denied, 7 CPU). Eight container metrics per service are aggregated to one minute, and seven trace features per service and minute are derived from the cross-service span join: mean and 95th-percentile inclusive and exclusive latency, error ratio, error-origin count, and throughput. The call graph is recovered from parent–child span links (Figure 2). Metric timestamps are in UTC, whereas spans and injection records use local time; both were aligned before processing. On nine of the 31 days some services have no trace records, so 28.3% of the incidents exhibit partial trace coverage.

SYN-50 is a multimodal benchmark for controlled cascading failures. Five topologies with 50 services each follow the structure reported for production call graphs: layered, tree-like graphs with heavy-tailed fan-out and stateless services calling stateful back-ends; four services share a host on average [2]. Per-minute metrics, traces, and logs are generated with latency and error propagation, retries, back-pressure that saturates the resources of affected callers, host contention, and random multiplicative spikes (×1.5–4) in 1% of all channel-minutes; 80% of the services are traced and 70% emit logs. Five fault types (CPU, memory, latency, disk, application error) are injected eight times per topology, yielding 200 incidents. Fault-free calibration windows are drawn from history for GAIA (300 windows without injections within ±70 minutes), from the normal period of each run for Sock-Shop, and from fault-free simulations for SYN-50.

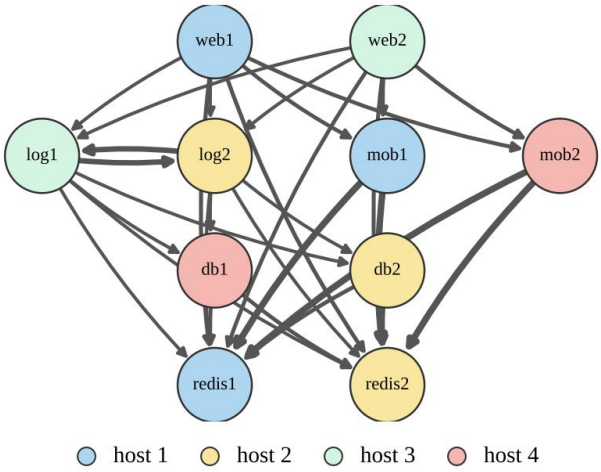


Figure 2 Call Layer of GAIA MicroSS Recovered from Parent–Child Span Links (Edge Width Proportional to Call Volume, Colors Denote Hosts)

Table 2 Datasets

Dataset	Services	Modalities	Types	Incidents
Sock-Shop [9]	13	metrics	2	50
GAIA [23]	10 (4 hosts)	metrics, traces	4	382
SYN-50	5 × 50	metrics, traces, logs	5	200

6.2 Baselines and Metrics

We compare with eight published methods: N-Sigma (robust 3σ rule over all channels), ε-Diagnosis, MicroRCA, CloudRanger, CIRCA (regression-based hypothesis testing on the structural graph derived from the call graph), RCD (official implementation, localized mode), TraceRCA, and MicroRank; the last two require traces and are not applicable to Sock-Shop [5, 8-10, 13-14, 17]. Baselines use the same windows and, where applicable, the same dependency graphs as DARE. We report the top-k accuracy AC@k, the average Avg@5 = $(1/5)\cdot\Sigma_{k=1}^{5}$ AC@k, and the runtime per incident. Recovery is evaluated in a closed loop: each incident is diagnosed by the respective method, and actions are executed in a simulator whose true action effectiveness deviates from the runbook prior by Gaussian noise with standard deviation 0.1, so that the planner operates under model mismatch. Each incident is replayed with 20 random seeds. We report the success rate within $K = 5$ actions, the mean time to recovery (MTTR, with a 30-minute escalation penalty for unrecovered incidents), the number of actions, and the disruption, defined as downtime weighted by $(1 + B(s))$. All DARE parameters are identical across datasets.

Table 3 Root Cause Localization Accuracy (Best Values in Bold; Runtime Measured on GAIA)

Method	Sock-Shop AC@1	AC@3	Avg@5	GAIA AC@1	AC@3	Avg@5	SYN-50 AC@1	AC@3	Avg@5	ms
N-Sigma	**1.000**	**1.000**	**1.000**	0.599	0.877	0.823	0.890	0.940	0.931	4.3
ε-Diagnosis	0.820	0.920	0.924	0.743	0.890	0.865	0.575	0.880	0.806	4.9

Method	Sock-Shop AC@1	AC@3	Avg@5	GAIA AC@1	AC@3	Avg@5	SYN-50 AC@1	AC@3	Avg@5	ms
MicroRCA	0.300	0.940	0.796	0.162	0.382	0.359	0.575	0.895	0.816	24.3
CloudRanger	0.180	0.260	0.272	0.039	0.285	0.333	0.195	0.520	0.485	50.4
CIRCA	0.820	0.920	0.924	0.736	**0.893**	0.865	0.425	0.755	0.686	17.5
RCD	0.140	0.300	0.304	0.709	0.866	0.839	0.285	0.500	0.458	423.6
TraceRCA	–	–	–	0.202	0.351	0.368	0.750	0.755	0.754	4.1
MicroRank	–	–	–	0.168	0.343	0.345	0.555	0.750	0.703	6.1
DARE (ours)	**1.000**	**1.000**	**1.000**	**0.817**	0.890	**0.883**	**0.930**	**0.960**	**0.957**	5.4

7 RESULTS

7.1 Root Cause Localization

Table 3 reports localization accuracy. DARE attains the highest top-1 accuracy on every dataset (shared with N-Sigma on Sock-Shop, where CPU and memory hogs dominate all other deviations): 1.000 on Sock-Shop, 0.817 on GAIA, and 0.930 on SYN-50. On GAIA, the strongest baseline is ε-Diagnosis with 0.743, so DARE improves AC@1 by 7.3 percentage points (9.9% relative); CIRCA and RCD reach 0.736 and 0.709. On SYN-50 the strongest baseline is N-Sigma with 0.890. Graph-walk methods are hampered by symptom-dominated graphs: MicroRCA (0.162 on GAIA) relies on response-time anomalies, which most memory faults do not produce, and CloudRanger (0.039) learns causal graphs from 70 samples per incident. Trace-only methods (TraceRCA 0.202, MicroRank 0.168) cannot see resource faults that do not alter request paths, which shows that span information is most valuable when it is fused with metrics as intrinsic evidence. RCD returns a small set of candidates and is effective when the injected metric dominates (GAIA), but it is less stable with per-second data that were collected days apart (Sock-Shop) and with hundreds of variables and few samples per incident (SYN-50: 0.285). TraceRCA is the strongest causal or trace baseline on SYN-50 (0.750), where faults alter request paths and traces cover 80% of the services. DARE diagnoses an incident in 5.4 ms on GAIA.
Figure 3 breaks the accuracy down by fault type. On GAIA, DARE localizes all access-denied faults (1.000): the errors propagate from the database-access services through the log services to the web services, and error-origin attribution isolates the database service. File-missing faults reach 0.947 and memory faults 0.817. In 45 of the 345 memory injections (13.0%) the root shows no memory deviation at all, which bounds the achievable accuracy; several CPU injections in the July release last only about three seconds, far below the one-minute aggregation interval. On SYN-50, the largest gains appear for faults that propagate to callers: latency faults (0.850 versus 0.750 for N-Sigma) and application errors (1.000 versus 0.900).

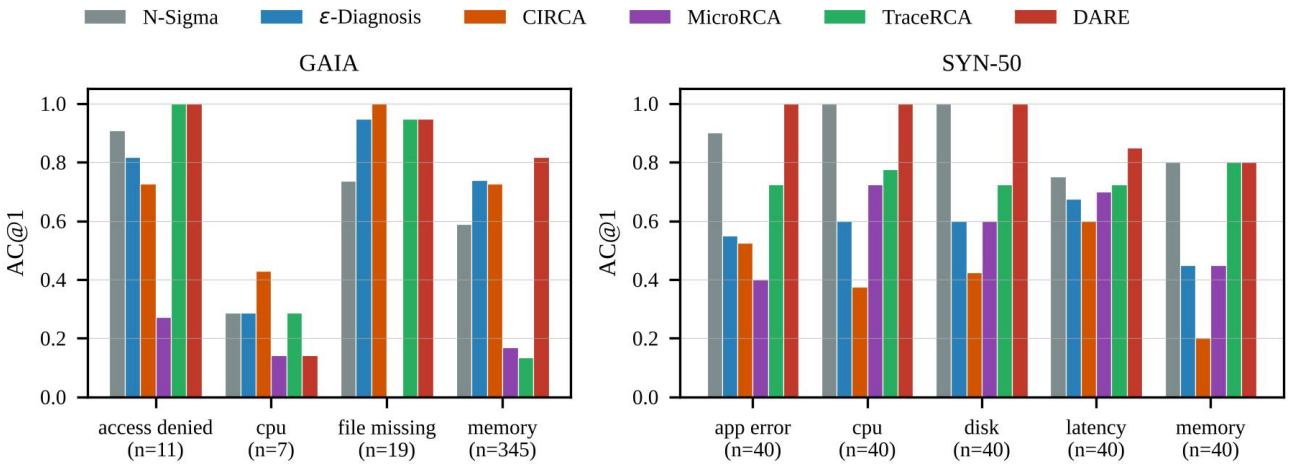


Figure 3 Top-1 Accuracy per Fault Type on GAIA and SYN-50 (n: Number of Incidents)

7.2 Ablation Study

Figure 4 removes one component at a time. History calibration is decisive on GAIA, whose channels are heterogeneous and noisy: without it, AC@1 drops by 39.0 points to 0.427, because network counters and tail latencies otherwise dominate the ranking. Trace-intrinsic evidence (exclusive latency and error origin) contributes 8.0 points on SYN-50, latent-state inference contributes 5.0 points on SYN-50, where 20% of the services are uninstrumented, and the propagation likelihood contributes 10.0 points on Sock-Shop and 2.0 points on SYN-50. The intrinsic-magnitude term separates hypotheses that explain the same symptoms (1.8 points on GAIA). Symptom gating keeps the top-1 accuracy within one point and raises Avg@5 on both trace-equipped datasets (0.957 versus 0.951 on SYN-50), and the host layer is neutral on these datasets, where co-location effects are weak relative to the injected faults.

7.3 Fault-Type Inference

Table 4 compares specificity-first typing with the naive choice of the channel with the largest raw deviation, both evaluated at the true root, as well as the end-to-end accuracy of DARE (root and type both correct). Specificity-first typing reaches 100.0% on Sock-Shop and 84.8% on GAIA, against 50.0% and 78.5% for the naive rule, and both rules perform on par on SYN-50 (92.0% and 92.5%), whose injected faults have mild cross-resource side effects. On Sock-Shop, every memory hog also drives CPU utilization close to one core, so the naive rule mislabels all memory faults as CPU faults; the typed belief therefore matters directly for action selection. End-to-end, DARE identifies both the root and the type for 78.8% of the GAIA incidents.

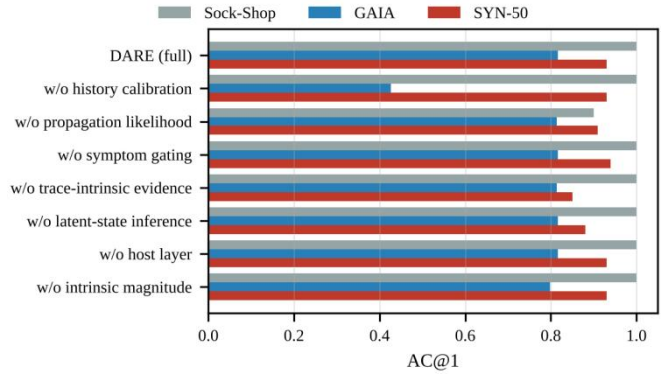


Figure 4 Ablation of DARE Components (AC@1)

Table 4 Fault-Type Inference Accuracy (%)

Dataset	Max-deviation channel	DARE (specificity-first)	DARE end-to-end
Sock-Shop	50.0	**100.0**	100.0
GAIA	78.5	**84.8**	78.8
SYN-50	**92.5**	92.0	89.0

7.4 Closed-Loop Recovery

Table 5 and Figure 5 report the closed-loop results. Restart-based remediation, the common practice, recovers 76.5% of the GAIA incidents and only 41.0% of the SYN-50 incidents within five actions, because latency, disk, and application faults are rarely cured by restarts. A fault-type-aware runbook driven by DARE raises the success rate on SYN-50 to 78.5%. DARE-Heal reaches 86.7% on GAIA, 95.0% on SYN-50, and 99.1% on Sock-Shop, and reduces the MTTR from 10.94 to 7.76 minutes on GAIA (−29.1%) and from 23.40 to 4.94 minutes on SYN-50 (−78.9%), with fewer actions. Its MTTR lies within 2.03 minutes of the oracle planner with perfect diagnosis on SYN-50. Belief refinement matters when the first attempt fails: without it, the success rate falls to 89.3% on SYN-50 and 83.9% on GAIA. The blast-radius term lowers the disruption imposed on dependent services by 8.7% on GAIA and 18.7% on SYN-50 compared with the planner without it, at nearly identical MTTR.

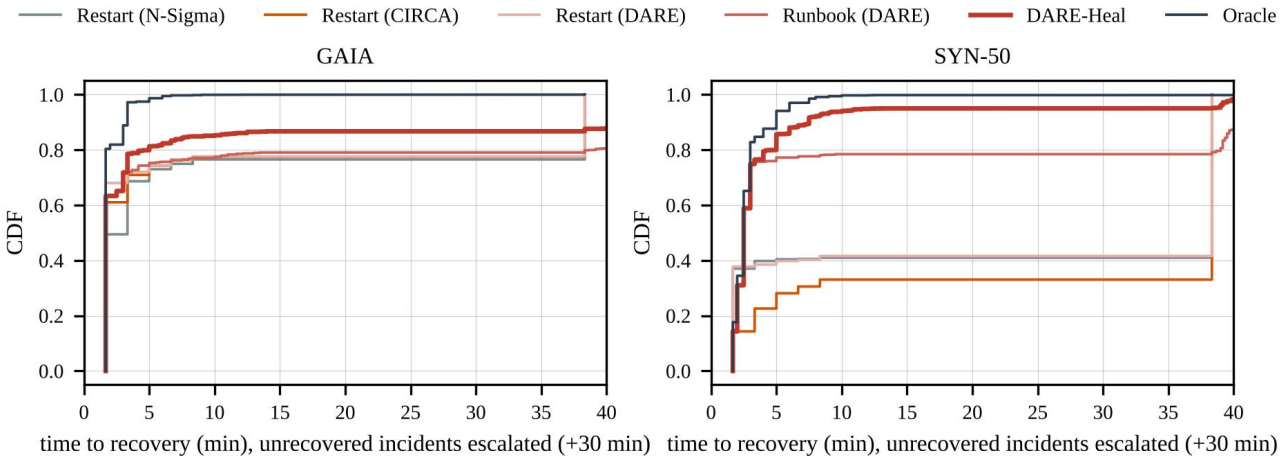


Figure 5 Distribution of the Time to Recovery over All Incidents and 20 Trials Each

Note: Incidents not recovered within five actions are escalated with a 30-minute penalty.

Table 5 Closed-Loop Recovery: Success Rate (%), MTTR (min), and Actions per Incident

Strategy	GAIA Succ.	MTTR	Act.	SYN-50 Succ.	MTTR	Act.	Sock-Shop Succ.	MTTR	Act.
Restart (N-Sigma)	76.5	10.94	2.34	41.0	23.40	3.42	73.1	11.53	2.08
Restart (CIRCA)	77.6	10.35	2.17	33.1	26.82	4.04	73.1	11.89	2.29

Strategy	GAIA Succ.	MTTR	Act.	SYN-50 Succ.	MTTR	Act.	Sock-Shop Succ.	MTTR	Act.
Restart (DARE)	77.6	10.19	2.09	41.6	23.24	3.44	73.1	11.53	2.08
Runbook (DARE)	79.0	10.70	1.98	78.5	10.62	1.94	86.8	6.70	1.53
DARE-Heal w/o belief update	83.9	8.82	1.91	89.3	7.02	1.70	98.4	2.81	1.23
DARE-Heal w/o blast radius	87.0	7.67	1.76	94.5	5.03	1.51	99.1	2.50	1.22
DARE-Heal	86.7	7.76	1.77	95.0	4.94	1.54	99.1	2.51	1.22
Oracle	100.0	2.02	1.13	99.8	2.91	1.23	99.9	2.19	1.17

7.5 Scalability and Sensitivity

Figure 6 scales SYN to 200 services (two topologies and 40 incidents per size; the three slowest baselines, MicroRCA, CloudRanger, and RCD, are evaluated on every fourth incident at 100 services and on every eighth incident at 200 services). DARE keeps an AC@1 of 0.875 at 200 services and needs 220 ms per incident, whereas CloudRanger and RCD, which learn causal graphs over all variables, require 52.2 s and 5.9 s, respectively. Figure 7 varies the main parameters around their defaults ($\rho_{up} \in [0.5, 0.9]$, $\tau \in [2, 5]$, $\lambda \in [0.5, 4]$, $\eta \in [0.1, 0.5]$). Across all tested values, AC@1 changes by at most 0.5 points on GAIA and 3.0 points on SYN-50; on Sock-Shop only the extreme settings $\tau = 2$ and $\lambda = 4$ lower AC@1 to 0.900 and 0.900. The defaults therefore lie in a broad stable region, and one configuration serves all three systems.

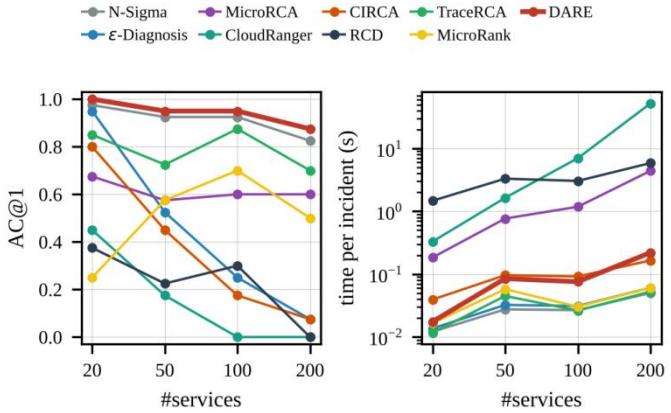


Figure 6 Accuracy and Runtime Versus System Size on Synthetic Topologies

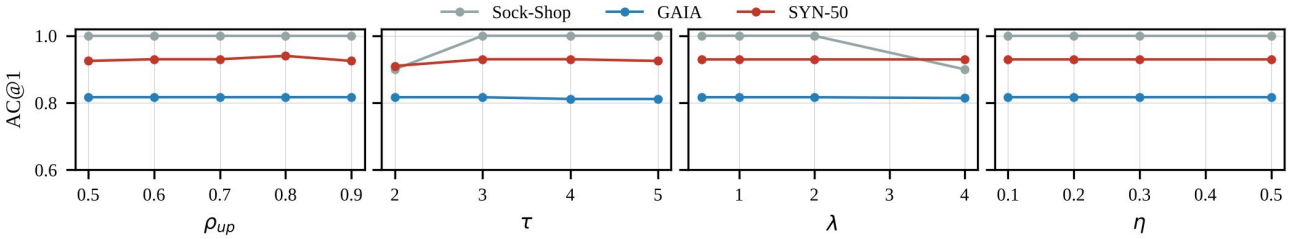


Figure 7 Sensitivity of AC@1 to the Propagation Decay ρ_{up}, the Threshold τ, the Magnitude Weight λ, and the Contention Probability η

8 DISCUSSION

DARE is designed for direct deployment. Calibration needs only fault-free history, which every monitored system accumulates; dependency graphs come from tracing or deployment descriptors; and the runbook priors are short, human-readable tables that operators can adapt to their platform, for example to add traffic shifting or feature-flag rollback. The verification step of DARE-Heal reuses the diagnosis evidence, so the same models that localize the fault also confirm its removal. Two extensions follow naturally from the probabilistic formulation: hypotheses with several concurrent roots can be scored by the same likelihood ratio over sets of services, and the action-effectiveness table can be updated online from verification outcomes, which turns DARE-Heal into a continually learning remediation engine.

9 CONCLUSION

This paper presented DARE, a dependency-aware framework for root cause analysis and automated recovery of cloud-native microservices. History-calibrated multimodal evidence, span-level intrinsic attribution, latent states for uninstrumented services, and a symptom-gated, typed propagation likelihood over call and co-location dependencies yield accurate and fast diagnoses; DARE-Heal turns the resulting belief into fault-specific, disruption-aware recovery actions and refines it after every attempt. Experiments on GAIA, Sock-Shop, and a multimodal cascading-failure

benchmark show that DARE matches or exceeds eight published baselines on every dataset, with clear margins on the two harder datasets, and that closed-loop recovery shortens the mean time to recovery by 29.1%–78.9% compared with restart-based remediation.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

REFERENCES

[1] Soldani J, Brogi A. Anomaly detection and failure root cause analysis in (micro)service-based cloud applications: A survey. ACM Comput. Surv., 2022, 55(3): 1-39.
[2] Luo S, Xu H, Lu C, et al. Characterizing microservice dependency and performance: Alibaba trace analysis. In: Proc. ACM Symp. Cloud Computing (SoCC), 2021: 412-426.
[3] Gan Y, Zhang Y, Hu K, et al. Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices. In: Proc. ASPLOS, 2019.
[4] Gan Y, Liang M, Dev S, et al. Sage: Practical and scalable ML-driven performance debugging in microservices. In: Proc. ASPLOS, 2021.
[5] Wang P, Xu J, Ma M, et al. CloudRanger: Root cause identification for cloud native systems. In: Proc. IEEE/ACM CCGRID, 2018: 492-502.
[6] Meng Y, Zhang S, Sun Y, et al. Localizing failure root causes in a microservice through causality inference. In: Proc. IEEE/ACM IWQoS, 2020: 1-10.
[7] Ma M, Xu J, Wang Y, et al. AutoMAP: Diagnose your microservice-based web applications automatically. In: Proc. Web Conf. (WWW), 2020: 246-258.
[8] Wu L, Tordsson J, Elmroth E, et al. MicroRCA: Root cause localization of performance issues in microservices. In: Proc. IEEE/IFIP NOMS, 2020.
[9] Ikram A, Chakraborty S, Mitra S, et al. Root cause analysis of failures in microservices through causal discovery. In: Proc. NeurIPS, 2022.
[10] Li M, Li Z, Yin K, et al. Causal inference-based root cause analysis for online service systems with intervention recognition. In: Proc. ACM SIGKDD, 2022.
[11] Spirtes P, Glymour C. An algorithm for fast recovery of sparse causal graphs. Soc. Sci. Comput. Rev., 1991, 9(1): 62-72.
[12] Sigelman B H, Barroso L A, Burrows M, et al. Dapper, a large-scale distributed systems tracing infrastructure. Google, Tech. Rep. dapper-2010-1, 2010.
[13] Li Z, Chen J, Jiao R, et al. Practical root cause localization for microservice systems via trace analysis. In: Proc. IEEE/ACM IWQoS, 2021.
[14] Yu G, Chen P, Chen H, et al. MicroRank: End-to-end latency issue localization with extended spectrum analysis in microservice environments. In: Proc. Web Conf. (WWW), 2021: 3087-3098.
[15] Liu D, He C, Peng X, et al. MicroHECL: High-efficient root cause localization in large-scale microservice systems. In: Proc. ICSE-SEIP, 2021: 338-347.
[16] Heckerman D, Breese J S, Rommelse K. Decision-theoretic troubleshooting. Commun. ACM, 1995, 38(3): 49-57.
[17] Shan H, Chen Y, Liu H, et al. ε-Diagnosis: Unsupervised and real-time diagnosis of small-window long-tail latency in large-scale microservice platforms. In: Proc. Web Conf. (WWW), 2019.
[18] Li Z, Zhao N, Li M, et al. Actionable and interpretable fault localization for recurring failures in online service systems. In: Proc. ESEC/FSE, 2022: 996-1008.
[19] Bahl P, Chandra R, Greenberg A, et al. Towards highly reliable enterprise network services via inference of multi-level dependencies. In: Proc. ACM SIGCOMM, 2007: 13-24.
[20] Kephart J O, Chess D M. The vision of autonomic computing. Computer, 2003, 36(1): 41-50.
[21] Candea G, Kawamoto S, Fujiki Y, et al. Microreboot—A technique for cheap recovery. In: Proc. USENIX OSDI, 2004.
[22] Weaveworks. Sock Shop: A microservices demo application. Available: https://github.com/microservices-demo/microservices-demo.
[23] CloudWise. GAIA: Generic AIOps Atlas dataset (MicroSS), version 1.10, 2022. Available: https://github.com/CloudWise-OpenSource/GAIA-DataSet.