ONLINE ROOT CAUSE ANALYSIS VIA SELF-GATING DUAL-BRANCH CAUSAL MECHANISM TRACKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reliable microservice operation hinges on three telemetry-monitoring tasks: anomaly detection, root cause analysis (RCA), and drift adaptation. All three concern the causal mechanisms generating the stream and must run online, on data already containing the faults to flag. Existing methods sidestep this setting by 1) consuming an anomaly timestamp, 2) assuming a clean training period, or 3) trusting an upstream detector, so faults silently pollute the learned reference. We show this pollution is invisible to detection yet fatal to localization: the reference absorbs exactly the contaminated fraction of each shift, shrinking the localization margin while the detection contrast stays positive. We therefore propose GDRC, a self-gating dual-branch causal mechanism tracker. Each window flows through two branches over a shared causal graph. The invariant branch maintains the reference of normal mechanisms; the variant branch refits the window as a sparse mechanism shift. A self-gating sampler compares the two by a BIC-penalized likelihood ratio, yielding per-node gains and a window responsibility. This responsibility gates the reference update, requiring no labels, no clean period, and no upstream detector. One framework serves all three tasks: detection, localization, and drift are read off the gains, so no timestamp is needed and no module absorbs another's errors. Experiments on controlled streams and two real benchmarks show that GDRC achieves competitive root-cause accuracy. Run end to end on a stream, it raises its own alarm and ranks there nearly as well as at the true fault onset. The same gain absorbs real traffic-regime changes while catching the faults that follow. Code and results are in the supplementary material.

1 INTRODUCTION

Keeping microservice systems reliable hinges on three telemetry-monitoring tasks commonly studied separately: anomaly detection, root cause analysis (RCA), and drift adaptation (Pham et al., 2024b; Gama et al., 2014). Yet all three concern one object, the causal mechanisms generating the telemetry (Pearl, 2009; Peters et al., 2017): an anomaly is a transient sparse mechanism shift (Schölkopf et al., 2021), its root cause is the shifted node, and drift is the same shift made permanent. One online mechanism model could therefore answer all three, provided its reference stays clean while learning from a stream that already contains the faults it must flag.

The obstacle is circular: a clean reference must learn only from normal windows, yet deciding which windows are normal is the detection problem itself. Existing methods sidestep the circle in three ways. Snapshot RCA methods (Ikram et al., 2022; Li et al., 2022; Budhathoki et al., 2022) consume the anomaly timestamp, and traversal methods (Chen et al., 2014; Lin et al., 2018; Ma et al., 2020) fire on a front-end threshold. Both skip the reference entirely, so adaptation lies outside their interface. Online approaches assume a clean training period, which production streams rarely offer because faults arrive unannounced. Other methods delegate windowing to an upstream detector, and every mistake of that detector enters the reference. In each case anomalous windows leak into the reference, and nothing in the pipeline reports how far the reference has moved. To our knowledge, the cost of this absorbed pollution has not been measured for any of the three tasks.

Our key observation is that the cost is sharply asymmetric; we call this failure mode reference pollution. On controlled structural causal model (SCM) streams at 40% contamination, an ungated reference still detects anomalies at AUC 0.991 while root-cause top-1 collapses from 0.830 to 0.401.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

A polluted reference partially absorbs the shifted mechanism, so residual evidence that should localize the cause spreads onto downstream symptom nodes. Detection needs only aggregate deviation, which pollution preserves; localization is exactly what it destroys. Crucially, the harm is multiplicative in shift magnitude and contamination, and it remains invisible to AUC.

To address this, we propose GDRC, a self-gating dual-branch causal mechanism tracker. In the offline stage, the causal graph is fixed from the service topology when one is available and from healthy data otherwise. In the online stage, each incoming window flows through two branches that share this graph. The invariant branch maintains the reference of normal behavior, a ridge regression of each node on its causal parents with exponential forgetting. The variant branch refits the current window as a sparse mechanism shift from the invariant parameters, modeling a fault as a few rewritten mechanisms. A self-gating sampler then compares the two explanations through a likelihood ratio carrying a Bayesian information criterion (BIC) penalty. The penalty charges every active shift coefficient, so the richer variant branch wins only when a concrete shift explains the window better than the reference. The comparison yields a per-node gain and the window's responsibility of being normal, computed in a single pass.

The responsibility closes the loop: it gates the invariant update without labels, so suspicious windows barely touch the reference, and every rejected window is instead explained by the variant branch. One framework thus serves all three tasks at once: the gain's magnitude detects, its location attributes, and its persistence separates drift from transient faults.

Experiments on two real benchmarks, PetShop (Hardt et al., 2024) and RCAEval RE1 (Pham et al., 2025), show that GDRC is competitive overall and the strongest causal RCA method on PetShop. Unlike timestamp-consuming baselines, it also raises its own alarms and absorbs real traffic-regime changes that break a static reference. We also report where the method falls short. BARO remains stronger under RE1's curated protocol, a dedicated change detector beats us at detection alone there, and diagnosing why yields a finding about the benchmark.

Our contributions are summarized as follows:

- We show that reference pollution breaks RCA but not detection, with multiplicative harm (Propositions 1 and 2, Section 4.4; verified in Section 5.2).
- We propose GDRC, a dual-branch self-gating sampler that breaks the circularity by routing each window against an explicit sparse-shift alternative, label-free after warmup (Section 4).
- We unify detection, RCA, and drift adaptation as read-offs of one gain and test this end to end on real streams (Section 5.2). The monitor raises its own alarm and ranks there almost as accurately as at the true fault onset. It also absorbs real traffic-regime changes while catching the faults that follow.
- We provide a boundary analysis: BARO leads on RE1, the deficit traces to topology, and RE1 rankings depend on input encoding more than algorithm choice (Section 5.3).

2 Related Work

Root cause analysis. Causal RCA localizes faults on a causal graph. CauseInfer, Microscope, and AutoMAP traverse the call graph with per-node detectors or random walks (Chen et al., 2014; Lin et al., 2018; Ma et al., 2020). RCD learns local structure around an anomaly indicator (Ikram et al., 2022). CIRCA (Li et al., 2022) and Assaad et al. (2023) test mechanism hypotheses on a given graph. Budhathoki et al. (2022) compute counterfactual contributions in a fitted SCM, and Han et al. (2025) score the recovered exogenous terms of a learned Granger model. Associational scorers such as ϵ-diagnosis (Shan et al., 2019) and BARO (Pham et al., 2024a; Adams & MacKay, 2007) remain strong on RCAEval (Pham et al., 2024b; 2025) and, in our runs, on PetShop (Hardt et al., 2024). All of these localize once. Given the anomaly timestamp, or a front-end alarm for the traversal methods, they compare windows before and after it, so the reference comes from outside. Two online methods come closer to our setting. CORAL (Wang et al., 2023) relearns the causal graph incrementally once a change detector fires, so the faulty batches it then learns from enter the graph. Yang et al. (2022) detect and attribute from one SCM fitted on a training set assumed clean.

Both leave the reference ungated. GDRC differs on exactly this point: it maintains the reference itself and gates every update, so it detects, localizes, and adapts from one penalized gain.

Causal discovery. Causal discovery learns the graph from data, and the methods fall into five families. Constraint-based methods such as PC and FCI test conditional independences (Spirtes et al., 2000). Score-based methods search graphs by a penalized score, greedily as in GES (Chickering, 2002) or by continuous optimization as in NOTEARS (Zheng et al., 2018). Functional methods such as LiNGAM exploit non-Gaussian noise to orient edges (Shimizu et al., 2006). Time-series methods add lags: Granger tests (Granger, 1969), PCMCI (Runge et al., 2019), DYNOTEARS (Pamfil et al., 2020), and Rhino with history-dependent noise (Gong et al., 2023). Interventional methods such as DCDI and ENCO use interventions to resolve what observational data leaves ambiguous (Brouillard et al., 2020; Lippe et al., 2022). All five families assume samples from a single, uncontaminated regime. Our stream mixes normal and faulty windows, so running discovery on it would reproduce the circularity we set out to break. Section 5.3 also shows that a wrong graph performs worse than an empty one. GDRC therefore takes the causal graph as an input, from the service topology or a healthy-period estimate, and spends its capacity on tracking the mechanisms over it.

Distribution shift. We instantiate the sparse mechanism shift hypothesis (Schölkopf et al., 2021; Perry et al., 2022) over an SCM (Pearl, 2009; Peters et al., 2017) in an online, contaminated stream. Several methods attribute a shift between distributions that are already separated. Budhathoki et al. (2021) explain the change between two given distributions, and iSCAN (Chen et al., 2023) localizes mechanism shifts across given environments. Our problem lies one step upstream, in deciding which windows define the reference in the first place. Our invariant branch is a temporal analogue of invariant causal prediction (Peters et al., 2016). Robust online estimation down-weights suspect samples one at a time, and concept drift methods adapt once the distribution has moved but leave the cause unexplained (Gama et al., 2014). Change detectors such as BOCPD, ADWIN, Page–Hinkley, and kernel statistics (Adams & MacKay, 2007; Bifet & Gavaldà, 2007; Page, 1954; Truong et al., 2020) likewise flag a transition without a reference to rank against. Our sampler instead routes whole windows against an explicit sparse-shift alternative, so standard ingredients (Schwarz, 1978; Tibshirani, 1996) keep the reference clean and attribute the shift.

3 Problem Setting

A monitor watches a microservice system whose n components each report a telemetry metric, and must decide whether the system is behaving abnormally, which component is responsible, and whether normal behavior itself has changed. The metrics arrive in windows $X^{(t)} \in \mathbb{R}^{T\times n}$ of T samples. In a healthy window, each metric is generated from its parents $\mathrm{Pa}(j)$ in a causal graph by a fixed function f_j, its *mechanism*: $X_j = f_j(\mathrm{Pa}(j)) + \varepsilon_j$ (Pearl, 2009; Peters et al., 2017). The healthy mechanisms $\{f_j\}$, collectively the *reference*, specify how every metric normally behaves. A fault such as a memory leak makes a window *anomalous*: it rewrites the mechanisms of a few nodes S into $\tilde{f}_j \neq f_j$, and other metrics move only by propagation (Schölkopf et al., 2021). A permanent rewrite, such as a lasting traffic change, is *drift*: the reference itself must move. The monitor is the online map $(X^{(1)}, \ldots, X^{(t)}) \mapsto (a^{(t)}, \rho^{(t)}, \hat{f}^{(t)})$. Since the true mechanisms are never given, $\hat{f}^{(t)}$ is the monitor's own running estimate of the reference, and every window is judged against it. The score $a^{(t)}$ should fire on anomalous windows (detection), the ranking $\rho^{(t)}$ should send the engineer to S first (localization), and $\hat{f}^{(t)}$ should follow drift but ignore faults (adaptation). Neither a window's status nor S is revealed; the monitor gets no anomaly timestamps, no upstream detector, and no labels beyond a trusted warmup.

4 Dual-Branch Online Causal Mechanism Tracking

4.1 Framework Overview

Figure 1 gives an overview of GDRC. In the offline stage, we fix the causal graph, from the service topology when available and from healthy data otherwise, and initialize the reference on the trusted warmup period. In the online stage, each window flows through the two branches sharing this

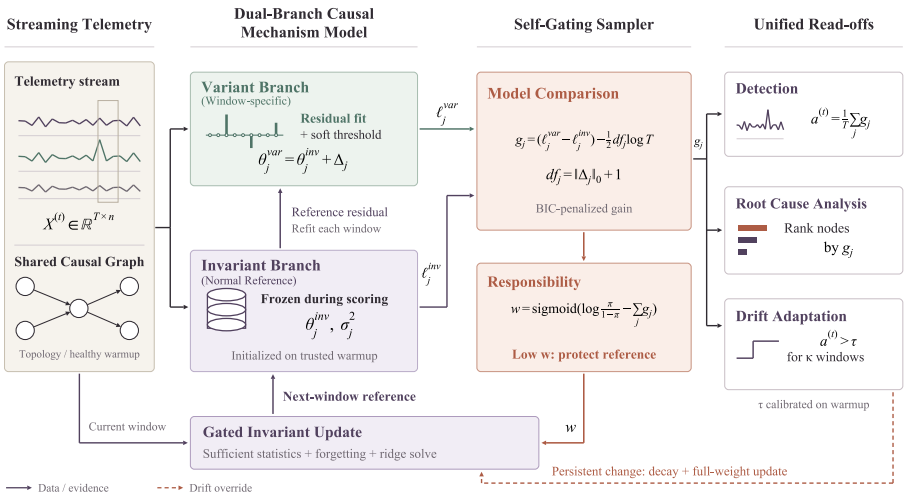

Figure 1: Overview of GDRC. Offline, the causal graph comes from the service topology or healthy data and a trusted warmup initializes the reference. Online, the invariant branch maintains the reference, the variant branch refits each window as a sparse mechanism shift, and the self-gating sampler compares them by a BIC-penalized likelihood ratio; the responsibility w gates the invariant update and the gains g_j serve all three read-offs.

graph: the *invariant* branch maintains the reference estimate $\hat{f}^{(t)}$, and the *variant* branch refits the window as a sparse mechanism shift. A self-gating sampler then compares the two explanations by a BIC-penalized likelihood ratio, yielding per-node gains g_j and a window responsibility w. The responsibility feeds back to gate the invariant update, so model comparison, not labels, decides which windows may update the reference. This decision is itself the detection problem, so the feedback breaks the circularity. The gains are read off as detection score, root-cause ranking, and drift signal, so all three tasks share one quantity and no module absorbs another's errors.

4.2 Offline Stage: Fixing the Causal Graph

A microservice system works by cooperation: each service does its part and relies on the services it calls. We represent this cooperation as a causal graph whose nodes are the services' metrics, in which an edge $i \to j$ means that metric j directly depends on metric i; A denotes the graph and $\mathrm{Pa}(j)$ the parents of j. The graph has two sources. When the platform records which service calls which, we reverse every edge of the call graph, because a slow callee slows its callers, so the callee is the parent. When no call graph is available, we instead build the graph from the healthy warmup data by giving each metric the k metrics most correlated with it as parents; Section 5.3 measures what this fallback costs. We fix the graph before the online stage, and it is the input to the two branches below, which learn how each metric depends on its parents but never change the edges.

4.3 Online Stage: Self-Gated Causal Mechanism Tracking

Invariant and variant branches. In the online stage, the stream mixes normal and anomalous windows, and the monitor must sort them. Every window asks two questions: what is normal, and what is happening now? A single adaptive model cannot answer both. To define normal behavior it must hold still, and to explain the current window it must move. We therefore split the roles across two branches. Both inherit the offline causal graph. The invariant branch owns the reference and moves slowly. The variant branch owns the current window and moves freely.

The invariant branch maintains the reference. Each node j regresses on its parents through a fixed feature map ϕ_j, $X_j = \theta_j^\top \phi_j(\mathrm{Pa}(j)) + \varepsilon_j$, with coefficients θ_j and Gaussian noise ε_j of variance σ_j^2. For linear mechanisms ϕ_j is the identity. For nonlinear ones it appends a fixed basis, such as $\tanh$ of each parent. The model stays linear in θ_j, so any fixed basis leaves the fit a ridge solve. The branch keeps exponentially forgotten sufficient statistics of this regression. The trusted warmup

period fills them first: its windows enter with full weight and yield the starting coefficients and noise variances. Starting from scratch fails: against an uninformed reference the variant branch wins every comparison, the responsibility pins near zero, and the reference starves. After warmup, each window enters the statistics weighted by the responsibility w that the sampler below assigns it. A ridge solve then recovers the reference coefficients θ_j^{inv} and noise variance σ_j^2, which together form the reference estimate $\hat{f}^{(t)}$. A window judged anomalous therefore enters with weight near zero and leaves the reference untouched.

The variant branch explains the window at hand, starting fresh on every window. When a window arrives, the invariant branch is frozen, and each node's reference prediction leaves a residual $r_j = X_j - \theta_j^{\mathrm{inv}\top}\phi_j(\mathrm{Pa}(j))$. The variant branch regresses this residual on the same parent features. A ridge step gives a shift estimate, and a soft-threshold prox step (Tibshirani, 1996) zeroes its small entries, yielding a sparse shift Δ_j. The variant mechanism is then $\theta_j^{\mathrm{var}} = \theta_j^{\mathrm{inv}} + \Delta_j$. Sparsity encodes the sparse mechanism shift hypothesis (Schölkopf et al., 2021). Because the shift lives in the reference's own parent coordinates, Δ_j names which of node j's incoming mechanisms moved and by how much. Each window thus exits with two explanations of the same data, the reference as it stands or the reference plus a sparse shift, and the decision between them falls to the sampler.

Self-gating sampler. The sampler decides which explanation the window earns, and with it whether the window may update the reference. This decision is the detection problem itself: admitting an anomalous window pollutes the reference, and rejecting a normal one starves it. Labels are unavailable, so the sampler settles it from inside the model. It compares the two explanations the window already carries, and the outcome sets the window's weight in the reference. We call this self-gating because the evidence comes from the model's own branches. The sampler is named for selecting which windows enter the reference. A window is down-weighted only when a concrete shift model beats the reference, a stricter test than a large residual.

Concretely, let ℓ_j^{inv} and ℓ_j^{var} denote the Gaussian log-likelihoods of node j's window under the two branches, with θ_j^{inv} held fixed. The variant branch has more parameters and fits at least as well, so the raw difference always favors it. We charge its extra freedom with a BIC penalty (Schwarz, 1978):

$$g_j \;=\; \big(\ell_j^{\mathrm{var}} - \ell_j^{\mathrm{inv}}\big) \;-\; \tfrac{1}{2}\,\mathrm{df}_j \log T, \qquad \mathrm{df}_j = \|\Delta_j\|_0 + 1, \tag{1}$$

where T is the window length and df_j counts the active shift coefficients plus the refit noise variance. A positive g_j means the shift at j survives its price, so g_j acts as a per-node significance test for a mechanism shift. Summing the same terms over nodes gives the window responsibility

$$w \;=\; \mathrm{sigm}\Big(\log \tfrac{\pi}{1-\pi} \;+\; \textstyle\sum_j \big(\ell_j^{\mathrm{inv}} - \ell_j^{\mathrm{var}} + \tfrac{1}{2}\,\mathrm{df}_j \log T\big)\Big), \tag{2}$$

where sigm is the logistic function and π the prior probability of a normal window, so w is the in-model posterior that the window is normal. Both quantities come from a single pass over the window. Since the alternative is refit on the very window it scores, we use w only as a routing score.

The responsibility then acts on both branches. The window joins the invariant statistics with weight w, closing the loop of Figure 1. A normal window thus keeps the reference current; an anomalous one enters with weight near zero. The windows the gate rejects are exactly those the variant branch fit, so data that would pollute the reference drives the ranking instead. Each window thus leaves the sampler with a responsibility w and per-node gains g_j, from which the three outputs are read off.

Three read-offs of one quantity. The gains g_j and the responsibility w that leave the sampler carry the monitor's three outputs. DETECTION. The anomaly score is the total gain per sample, $a^{(t)} = \sum_j g_j/T$, the penalized likelihood-ratio statistic of the whole window, and a threshold is needed only to raise an alarm. LOCALIZATION. The ranking $\rho^{(t)}$ orders nodes by g_j. As T grows, $\frac{1}{T}\,\mathbb{E}[g_j]$ converges to the Kullback–Leibler divergence between node j's conditionals under the two branches, so the ranking sends the engineer to the mechanism that moved furthest from the reference (Budhathoki et al., 2021). On real call graphs a soft ancestor prior can refine the order. ADAPTATION. Drift and faults look identical window by window, and without labels the only separating signal is duration: a change that persists is the new normal. When κ consecutive windows alarm, $a^{(t)} > \tau$, the invariant statistics decay by an extra factor α and the window enters with full weight. The threshold τ is the model-implied $\tau_\pi = \frac{1}{T}\log\frac{\pi}{1-\pi}$ (equivalent to $w < \frac{1}{2}$) when the

invariant branch is well specified, and is calibrated on the warmup on real telemetry (Section 5.2). A shorter fault leaves the streak incomplete and still fires the detector. A longer one is absorbed by design (Section 5.3). Each window costs one ridge solve per node, $O(\sum_j p_j^3 + T\sum_j p_j^2)$ with $p_j = |\mathrm{Pa}(j)|$, milliseconds for hundreds of nodes (Appendix A).

4.4 Why Pollution Breaks RCA but Not Detection

All three read-offs are judged against the reference, so pollution is the one failure that reaches every output. Two questions follow. How much of a shift does an ungated reference absorb, and why does the damage fall on RCA but spare detection? We answer both with a linear-Gaussian calculation for a single shifted node j^* with true coefficients θ^* and noise variance σ^2. A contamination fraction c of windows draws X_{j^*} from the shifted coefficients $\theta^* + \Delta$. Its parents are unshifted, so both window types share the covariate second moment C of ϕ_{j^*}.

Proposition 1 (Exact absorption). *In steady state the ungated invariant estimates satisfy $\hat{\theta} = \theta^* + c\,\Delta$ and $\hat{\sigma}^2 = \sigma^2 + c(1-c)\,q$, where $q = \Delta^\top C \Delta$ is the shift energy. A gated estimator whose responsibilities separate the two window types recovers θ^* and σ^2 for every $c < 1$.* The ungated reference absorbs exactly the contaminated fraction of the shift, with parameter error $c\,\|\Delta\|$ linear in contamination, linear in magnitude, and multiplicative in both. The gated estimator asks only that the responsibilities separate the two window types, so gating keeps the reference clean at any contamination short of a fully anomalous stream.

Proposition 2 (Detection–RCA asymmetry). *To leading order in q/σ^2, under the polluted reference the expected per-sample gain at j^* is $(1-c)^2 q/2\sigma^2$ on anomalous windows and $c^2 q/2\sigma^2$ on normal ones. Every unshifted node's expected gain sits at or below zero. The pooled detection contrast between anomalous and normal windows is therefore $(1-2c)\,q/2\sigma^2$, positive for every $c < \frac{1}{2}$. The localization margin at j^* shrinks as $(1-c)^2$ and is deflated further by the variance inflation of Proposition 1. At finite T it must additionally clear the largest of $n-1$ zero-mean fluctuations.* Detection compares whole windows, so the contrast stays positive up to one-half contamination; RCA compares nodes against a margin that pollution attacks from the numerator and the denominator at once, and at finite T the variant support also spreads onto descendants. Both propositions concern population means, so they predict directions and scalings, which Section 5.2 tests by removing the gate. Appendix B gives the proofs.

5 Experiments

5.1 Experimental Setup

Datasets. We use controlled synthetic streams and two real telemetry benchmarks. Synthetic streams come from linear and nonlinear 12-node SCMs (edge probability 0.3) with Gaussian noise. An anomaly is a sparse mechanism shift of magnitude s at one node (Schölkopf et al., 2021) that reaches downstream nodes only through propagation, and a fraction c of windows is anomalous. Windows have T=200 samples unless noted, thirty trusted warmup windows bootstrap the invariant branch, and every configuration runs over five seeds. PetShop (Hardt et al., 2024) has 41 components at 5-minute resolution, 68 injected issues (36 latency, 32 availability), and a call graph with edges running callee to caller. RCAEval RE1 (Pham et al., 2025; 2024b) has 375 failure cases from three microservice systems at $\sim$1 s resolution but no call graph; its causal graph uses each metric's top-3 correlation parents on the normal period.

Baselines. For RCA we compare against BARO (Pham et al., 2024a), CIRCA/HT (Li et al., 2022), RCD (Ikram et al., 2022), counterfactual attribution (Budhathoki et al., 2022), and a robust-z scorer, all run under the official harnesses with the authors' implementations (Pham et al., 2024a; Liu et al., 2023; Sharma & Kiciman, 2020). For detection we add four change detectors under the same warmup and windows: BOCPD (Adams & MacKay, 2007), ADWIN (Bifet & Gavaldà, 2007), Page–Hinkley (Page, 1954), and a windowed kernel change statistic (Truong et al., 2020). Each runs per metric and aggregates by max, as robust-z does. For drift the baseline is a static reference that never absorbs. BARO, CIRCA, and RCD consume the anomaly timestamp, so detection and drift lie outside their interface; the change detectors keep no reference, so RCA lies outside theirs.

Table 1: Contamination robustness on synthetic streams with gating (S) and without (noS), five seeds per configuration. Best per condition in **bold**.

	inv-err		detection AUC		RCA top-1	
contamination	S	noS	S	noS	S	noS
10%	**0.0081**	0.0149	**1.000**	1.000	**0.942**	0.830
20%	**0.0086**	0.0159	**1.000**	0.998	**0.968**	0.666
30%	**0.0081**	0.0193	**1.000**	0.995	**0.955**	0.589
40%	**0.0085**	0.0244	**1.000**	0.991	**0.976**	0.401

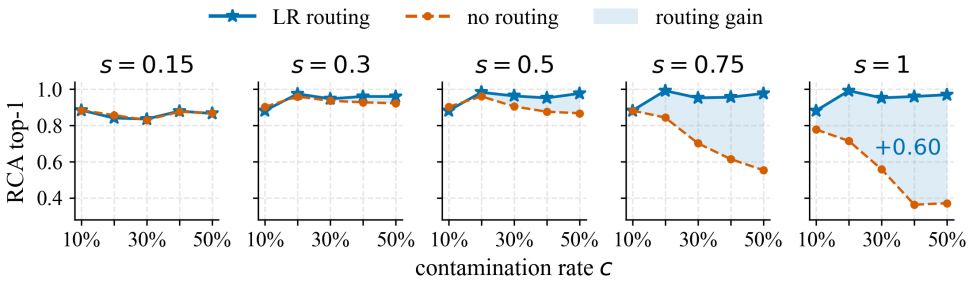

Figure 2: RCA top-1 with likelihood-ratio routing (solid) and without (dashed) as contamination grows, one panel per shift magnitude. Shading marks the routing gain.

Evaluation metrics. Synthetic metrics are inv-err (relative Frobenius error of the invariant weights), detection AUC, and RCA top-1. On real telemetry, RCA is top-1/top-3 (PetShop) and AC@1/AC@3 (accuracy at k; RE1). Detection is pooled average precision (AP) with ROC AUC alongside, since positives are only 0.9% (PetShop) and 13% (RE1) of windows. Drift is the false-alarm rate per window after a real regime change, paired with detection of the fault that follows.

5.2 Experimental Results

Reference pollution breaks RCA but not detection. This experiment measures what a polluted reference costs. We switch the gate off on synthetic streams with a known contamination rate. Table 1 reports the reference error, detection AUC, and RCA top-1 as contamination grows, with and without gating. Without gating top-1 halves while detection AUC stays above 0.99, and with gating both stay flat. A polluted reference therefore still detects faults but can no longer locate them, which is exactly the asymmetry that Propositions 1 and 2 predict.

Harm is multiplicative in shift magnitude and contamination. This experiment asks whether the damage depends on how large the fault is and on how many windows contain one. We sweep shift magnitude and contamination rate jointly on the same streams, with and without gating. Figure 2 plots RCA top-1 against contamination, one panel per shift magnitude. Without gating the curves fan downward as contamination grows, steeply for large shifts and not at all for small ones. With gating they stay flat in every panel, so the damage grows with both factors as predicted. A residual-threshold router or no router restores the collapse (Appendix C), so the explicit alternative model is what protects the reference.

Root cause analysis on real telemetry. This experiment asks whether the method finds the faulty component in real microservice systems. We run it and existing RCA methods on both benchmarks under the official protocol, which hands every method the faulty window. Table 2 reports top-k accuracy for all methods. Among causal methods GDRC is the strongest on PetShop, where the metric-aware graph prior drives the result. On RE1 it trails BARO and CIRCA, so the mechanism model localizes real faults yet does not dominate. Engineered associational scorers remain stronger on PetShop top-1, which we attribute to trigger-point search and tuned robust scoring, and Section 5.3 traces the RE1 gap.

Detection. This experiment asks whether the same anomaly score can also raise the alarm, a job normally left to a separate detector. We score it against five change detectors given the same warmup.

Table 2: Capability coverage on real telemetry. RCA: top-1 / top-3 (PetShop), AC@1 / AC@3 (RE1). Detection: pooled AP (ROC AUC) over $68/7444$ and $375/2500$ positives/negatives. Drift: post-switch false alarms per window / detection of the fault that follows (204 PetShop streams). Empty cells mark outputs a method does not produce. Best per column in **bold**.

	RCA		detection AP (AUC)		drift
method	PetShop	RE1	PetShop	RE1	FA / fault det.
BARO	n/a	**0.685 / 0.880**	n/a	n/a	n/a
CIRCA / HT	0.382 / **0.794**	0.533 / 0.696	n/a	n/a	n/a
RCD	0.265 / 0.309	0.237 / 0.301	n/a	n/a	n/a
robust-z	0.176 / 0.441	0.104 / 0.448	0.575 (0.803)	0.558 (0.746)	n/a
kernel change det.	n/a	n/a	0.535 (0.980)	**0.863** (**0.973**)	n/a
static reference (ablation)	–	–	–	–	0.574 / **0.882**
GDRC (ours)	**0.485 / 0.794**	0.347 / 0.691	**0.601** (**0.990**)	0.620 (0.837)	**0.258** / 0.877

Table 3: Autonomous operation on RE1 and PetShop. For each detected case (RE1 at margin 1, 338 of 375; PetShop 52 of 68) the same ranker is read at three references: our own alarm window (no timestamp), the true injection window, and the full abnormal stretch. Delay is the number of windows from injection to our alarm. The offline protocol row is the benchmark number of Table 2.

			AC@1			AC@3		
fault	n	delay	own alarm	injection	full stretch	own alarm	injection	full stretch
CPU	65	0.49	0.231	0.246	0.215	0.492	0.477	0.600
DELAY	67	0.15	0.179	0.179	0.164	0.373	0.388	0.403
DISK	70	0.50	0.229	0.271	0.086	0.600	0.571	0.557
LOSS	69	0.30	0.101	0.072	0.159	0.406	0.319	0.406
MEM	67	0.10	0.478	0.507	0.358	0.701	0.731	0.701
all RE1	338	0.31	**0.243**	**0.254**	**0.195**	**0.515**	**0.497**	**0.533**
offline (all RE1)	375	n/a	n/a	n/a	0.347	n/a	n/a	0.691
PetShop (top-1, top-3)	52	n/a	0.442	0.442	0.442	0.615	0.615	0.615

Table 2 reports AP and AUC on both benchmarks. On PetShop the score is the best detector, and marginal scorers match it only on availability faults. On latency faults they fall to a third of its AP, because conditioning on parents separates a propagated symptom from an unshifted node. On RE1 a windowed kernel change statistic is stronger, a loss traced in Section 5.3.

Autonomous operation. This experiment asks what happens when nobody tells the method when the fault occurred. Instead of receiving the faulty window, the model watches the whole stream, raises its own alarm from a warmup-calibrated threshold, and ranks at that alarm. Table 3 reports the ranking at our own alarm, at the true injection window, and on the full abnormal stretch. Our own alarm costs 0.011 AC@1 against the true injection window, so the timestamp every baseline receives is worth little. One short window after onset even beats the full abnormal stretch. The remaining gap to the offline protocol comes from the streaming model having seen far less normal behavior before the fault arrives.

Drift adaptation. This experiment asks whether the method can tell a harmless change in traffic from a fault. We build each stream from one PetShop traffic regime, cross into another, and then inject a fault, for three absorption rules. Table 4 reports post-switch false alarms and detection of the subsequent fault. A static reference alarms on most post-switch windows. The calibrated persistence rule halves these alarms and still catches the fault at the same rate, so absorbing the benign change costs no sensitivity. The model-implied threshold instead absorbs in every case and swallows some availability faults, which is why the threshold is calibrated on the warmup.

5.3 Analysis: Where the Method Falls Short, and Why

BARO is stronger on RE1. BARO is the strongest method on RE1 and leads on every system and fault type (paired tests in Appendix E). Figure 3 looks for the cause in three places. Panel (a) asks whether accuracy is lost when per-node scores are pooled into one score per service. Even an oracle told the fault's metric family stays well below BARO, so pooling is not the cause. Panel (b) points to

Table 4: Drift selectivity on 204 real PetShop regime changes, each followed by an injected fault. Drift FA: fraction of post-switch windows that alarm. Absorbed: fraction of cases in which the rule absorbed the regime change. Fault detected: fraction of subsequent faults that alarmed. Fault absorbed: fraction wrongly absorbed as drift.

metric	rule	drift FA	absorbed	fault detected	fault absorbed
availability (n=96)	static	0.708	0.000	0.938	0.000
	model-implied τ_π	0.292	1.000	0.812	0.125
	calibrated τ (ours)	0.354	0.583	0.938	0.000
latency (n=108)	static	0.454	0.000	0.833	0.000
	model-implied τ_π	0.173	1.000	0.870	0.000
	calibrated τ (ours)	0.173	0.389	0.824	0.000

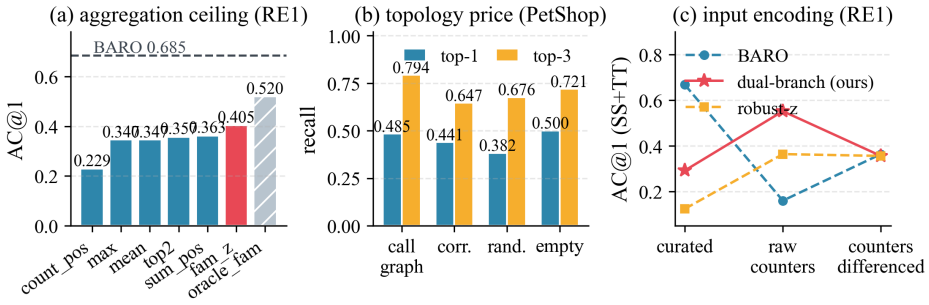

Figure 3: Anatomy of the RE1 gap. (a) Pooling reductions, even with a family oracle (hatched), stay below BARO. (b) The correlation graph costs 14.7pp of top-3 on PetShop. (c) On raw counters BARO collapses and our score rises, and differencing ties all three.

the graph. RE1 ships no call graph, so parents are estimated from correlations, and on PetShop that substitution costs 14.7 points of top-3. Panel (c) shows that RE1 rankings depend on how metrics are encoded. On raw cumulative counters BARO collapses and our score rises, and one differencing pass makes the three methods tie.

A change detector is stronger at RE1 detection. On RE1 a windowed kernel change statistic detects faults better than our score, both in the offline comparison and in the streaming one. That detector compares each window only with the window before it, so it never uses the causal graph. Our score conditions every node on its estimated parents, so the wrong parents that hurt attribution on RE1 hurt detection too. The two RE1 losses therefore share one cause, the correlation graph. The change detector also pays for its simplicity. It keeps no model of normal behavior, so it cannot rank root causes and cannot tell drift from a fault.

6 Conclusion

We proposed GDRC, a self-gating dual-branch tracker that keeps a clean causal reference on a stream that already contains the faults it must flag. Its invariant branch holds the reference, and its variant branch refits each window as a sparse mechanism shift. A BIC-penalized likelihood ratio between them gates the reference update and yields one per-node gain that serves detection, localization, and drift adaptation. The experiments leave four take-aways. Reference pollution is asymmetric, destroying localization while leaving detection intact, so detection metrics cannot certify an online RCA model. Its damage grows with both fault size and contamination, which makes it worst for the large, frequent faults that matter most. Gating against an explicit alternative model keeps the reference clean, and once the model raises its own alarm a fault timestamp adds almost nothing. On real benchmarks the quality of the causal graph decides both localization and detection, and a wrong graph is worse than none. Future work will replace the stationary Gaussian invariant branch with heavy-tailed likelihoods, so that the responsibility needs no warmup threshold. It will also learn the causal graph online from traces and let the streaming reference keep learning from the healthy windows that the gate admits.

AI USE STATEMENT

Generative AI tools were used in two ways. First, they polished the language of the manuscript: rewording, terminology consistency, and tightening to the page limit. Second, they searched for gaps in the evaluation, which led to the additional change-detection baselines, the end-to-end streaming evaluation, and the drift-selectivity suite. The authors ran all of these experiments. AI-assisted analysis scripts were executed by the authors, and every reported number was re-derived from the archived result files of Appendix G. The authors take full responsibility for all content, including artifacts produced with AI assistance.

REPRODUCIBILITY STATEMENT

Code, data-processing scripts, and the archived per-run result files behind every table and figure are provided as supplementary material. Appendix G maps each script to the table or figure it produces, lists the result-file inventory, and pins the software environments; Appendix B gives the proofs of Propositions 1 and 2; Appendices C and D specify the synthetic generator, the benchmark protocols, the baselines' official harnesses, and the seeds. Both benchmarks are public: PetShop (Hardt et al., 2024) under CC-BY-4.0 and the RE1 split of RCAEval (Pham et al., 2025) under Zenodo DOI 10.5281/zenodo.14590730. Hardware and software platform details, including the pinned environments of the baselines, are given in Appendix G.

REFERENCES

Ryan Prescott Adams and David J. C. MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

Charles K. Assaad, Imad Ez-Zejjari, and Lei Zan. Root cause identification for collective anomalies in time series given an acyclic summary causal graph with loops. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 206 of *Proceedings of Machine Learning Research*, pp. 8395–8404. PMLR, 2023.

Albert Bifet and Ricard Gavaldà. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pp. 443–448, 2007.

Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. Differentiable causal discovery from interventional data. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Kailash Budhathoki, Dominik Janzing, Patrick Blöbaum, and Hoiyi Ng. Why did the distribution change? In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130 of *Proceedings of Machine Learning Research*, pp. 1666–1674. PMLR, 2021.

Kailash Budhathoki, Lenon Minorics, Patrick Blöbaum, and Dominik Janzing. Causal structure-based root cause analysis of outliers. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2357–2369. PMLR, 2022.

Pengfei Chen, Yong Qi, Pengfei Zheng, and Di Hou. CauseInfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems. In *Proceedings of IEEE INFOCOM*, pp. 1887–1895. IEEE, 2014.

Tianyu Chen, Kevin Bello, Bryon Aragam, and Pradeep Ravikumar. iSCAN: Identifying causal mechanism shifts among nonlinear additive noise models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1–37, 2014.

Wenbo Gong, Joel Jennings, Cheng Zhang, and Nick Pawlowski. Rhino: Deep causal temporal relationship learning with history-dependent noise. In *International Conference on Learning Representations*, 2023.

Clive W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438, 1969.

Xiao Han, Saima Absar, Lu Zhang, and Shuhan Yuan. Root cause analysis of anomalies in multivariate time series through Granger causal discovery. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Michaela Hardt, William Roy Orchard, Patrick Blöbaum, Elke Kirschbaum, and Shiva Kasiviswanathan. The PetShop dataset — finding causes of performance issues across microservices. In *Proceedings of the Third Conference on Causal Learning and Reasoning (CLeaR)*, volume 236 of *Proceedings of Machine Learning Research*, pp. 957–978. PMLR, 2024.

Azam Ikram, Sarthak Chakraborty, Subrata Mitra, Shiv Kumar Saini, Saurabh Bagchi, and Murat Kocaoglu. Root cause analysis of failures in microservices through causal discovery. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 31158–31170, 2022.

Mingjie Li, Zeyan Li, Kanglin Yin, Xiaohui Nie, Wenchi Zhang, Kaixin Sui, and Dan Pei. Causal inference-based root cause analysis for online service systems with intervention recognition. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 3230–3240. ACM, 2022.

Jinjin Lin, Pengfei Chen, and Zibin Zheng. Microscope: Pinpoint performance issues with causal graphs in micro-service environments. In *International Conference on Service-Oriented Computing (ICSOC)*, volume 11236 of *Lecture Notes in Computer Science*, pp. 3–20. Springer, 2018.

Phillip Lippe, Taco Cohen, and Efstratios Gavves. Efficient neural causal discovery without acyclicity constraints. In *International Conference on Learning Representations*, 2022.

Chenghao Liu, Wenzhuo Yang, Himanshu Mittal, Manpreet Singh, Doyen Sahoo, and Steven C. H. Hoi. PyRCA: A library for metric-based root cause analysis. *arXiv preprint arXiv:2306.11417*, 2023.

Meng Ma, Jingmin Xu, Yuan Wang, Pengfei Chen, Zonghua Zhang, and Ping Wang. AutoMAP: Diagnose your microservice-based web applications automatically. In *Proceedings of The Web Conference (WWW)*, pp. 246–258. ACM, 2020.

Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet. River: machine learning for streaming data in Python. *Journal of Machine Learning Research*, 22(110):1–8, 2021.

E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. DYNOTEARS: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 108 of *Proceedings of Machine Learning Research*, pp. 1595–1605, 2020.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009.

Ronan Perry, Julius von Kügelgen, and Bernhard Schölkopf. Causal discovery in heterogeneous environments under the sparse mechanism shift hypothesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 10904–10917, 2022.

Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society: Series B*, 78(5):947–1012, 2016.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, 2017.

Luan Pham, Huong Ha, and Hongyu Zhang. BARO: Robust root cause analysis for microservices via multivariate Bayesian online change point detection. *Proceedings of the ACM on Software Engineering*, 1(FSE):2214–2237, 2024a. doi: 10.1145/3660805.

Luan Pham, Huong Ha, and Hongyu Zhang. Root cause analysis for microservice system based on causal inference: How far are we? In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 706–715. ACM, 2024b. doi: 10.1145/3691620.3695065.

Luan Pham, Hongyu Zhang, Huong Ha, Flora Salim, and Xiuzhen Zhang. RCAEval: A benchmark for root cause analysis of microservice systems with telemetry data. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion)*, pp. 777–780. ACM, 2025. doi: 10.1145/3701716.3715290.

Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science Advances*, 5 (11):eaau4996, 2019.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

Huasong Shan, Yuan Chen, Haifeng Liu, Yunpeng Zhang, Xiao Xiao, Xiaofeng He, Min Li, and Wei Ding. ϵ-diagnosis: Unsupervised and real-time diagnosis of small-window long-tail latency in large-scale microservice platforms. In *Proceedings of the World Wide Web Conference (WWW)*, pp. 3215–3222. ACM, 2019.

Amit Sharma and Emre Kiciman. DoWhy: An end-to-end library for causal inference. *arXiv preprint arXiv:2011.04216*, 2020.

Shohei Shimizu, Patrik O. Hoyer, Aapo Hyvärinen, and Antti Kerminen. A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7:2003–2030, 2006.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. MIT Press, 2nd edition, 2000.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996.

Charles Truong, Laurent Oudre, and Nicolas Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020.

Dongjie Wang, Zhengzhang Chen, Yanjie Fu, Yanchi Liu, and Haifeng Chen. Incremental causal graph learning for online root cause analysis. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 2269–2278. ACM, 2023.

Wenzhuo Yang, Kun Zhang, and Steven C. H. Hoi. A causal approach to detecting multivariate time-series anomalies and root causes. *arXiv preprint arXiv:2206.15033*, 2022.

Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. DAGs with NO TEARS: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, volume 31, pp. 9492–9503, 2018.

A METHOD DETAILS

Notation. Table 5 collects the symbols used in Section 4 and in Algorithm 1. The per-node gain g_j and the responsibility w are defined in equation 1 and equation 2.

Table 5: Notation.

symbol	meaning
$X^{(t)} \in \mathbb{R}^{T\times n}$	window t: T samples of n monitored variables
$A \in \{0,1\}^{n\times n}$	shared causal graph; $A_{ij}=1$ means i is a parent of j; $\mathrm{Pa}(j)$, $p_j = \lvert\mathrm{Pa}(j)\rvert$
$\phi_j(\cdot)$, Φ_j	feature map of node j (identity, or $[x, \tanh x]$ / $[x, \sin x]$) plus intercept; design matrix $\Phi_j \in \mathbb{R}^{T\times d_j}$
$\theta_j^{\mathrm{inv}}, \sigma_j^2$	invariant-branch coefficients and noise variance of node j
$\Delta_j \in \mathbb{R}^{d_j}$	variant-branch sparse shift; $\theta_j^{\mathrm{var}} = \theta_j^{\mathrm{inv}} + \Delta_j$, refit noise scale $\tilde\sigma_j^2$
$\ell_j^{\mathrm{inv}}, \ell_j^{\mathrm{var}}$	Gaussian log-likelihoods of node j's window residuals under the two branches
g_j	BIC-penalized gain, equation 1; $g_j > 0$ is the per-node significance test
df_j	$\lVert\Delta_j\rVert_0 + 1$; the $+1$ charges the refit of $\tilde\sigma_j^2$
$w \in [0,1]$	window responsibility, the in-model posterior equation 2, used as a routing score
π	prior probability of a normal window
W	PetShop scoring-window length in 5-minute timestamps (Table 15)
λ	soft-threshold level on Δ, in units of σ_j
ρ	exponential forgetting factor of the invariant statistics
κ	drift patience: consecutive alarm windows before absorption
τ	alarm threshold; $\tau_\pi = \frac{1}{T}\log\frac{\pi}{1-\pi}$ ($\equiv w < \frac{1}{2}$) or warmup-calibrated
α	absorption decay applied to the invariant statistics on drift
$S_{xx}, S_{xy}, S_n, S_{rr}$	per-node sufficient statistics: $\sum \rho^{\cdot} w\, \Phi^\top\Phi$, $\sum \rho^{\cdot} w\, \Phi^\top y$, effective sample count, and weighted residual sum of squares

Algorithm. Algorithm 1 gives the full per-window update. A single pass fits the sparse alternative and scores the comparison with θ^{inv} held fixed. The detection read-off, the routing arm, the drift rule, and the gated invariant update follow. The routing arm has three settings. The likelihood-ratio router is named `em` in the released code, `residual` replaces the responsibility with a residual-magnitude test, and `none` always uses weight 1. The released code also keeps an iteration flag `em_iters` (default 3) from an earlier design. Extra passes recompute identical quantities because θ^{inv} is fixed within the window, so the flag leaves every output unchanged.

Hyperparameter defaults. All experiments share the same defaults. The forgetting factor is $\rho = 0.98$ and the normal-window prior is $\pi = 0.7$. The soft-threshold level is $\lambda = 0.05$ on synthetic streams and $\lambda = 0.10$ on PetShop (sensitivity in Appendix D). Synthetic streams use drift patience $\kappa = 5$ with the model-implied threshold τ_π. The real drift suite runs on 5-sample PetShop windows with $\kappa = 3$, $\alpha = 0.5$, and a warmup-calibrated τ (margin 3). The invariant branch is bootstrapped on a trusted healthy period: 30 windows on synthetic streams, an early normal slice on the benchmarks. Gating from scratch never learns, since the variant branch always beats an uninformed invariant.

B POLLUTION BIAS AND THE DETECTION–RCA ASYMMETRY: PROOFS

This appendix proves Propositions 1 and 2 and states the separation condition behind the gated claim. Node j^* follows $y = \phi^\top\theta + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0,\sigma^2)$ and covariate second moment $C = \mathbb{E}[\phi\phi^\top] \succ 0$. The coefficients are θ^* on normal windows and $\theta^* + \Delta$ on anomalous ones, and each window is anomalous independently with probability c. The parents of j^* are unshifted, so C is the same on both window types, and $q = \Delta^\top C\Delta$ is the shift energy. We work in the population steady state of the exponentially forgotten statistics, so every expectation below is over the window-type mixture.

Proof of Proposition 1. The ungated update accumulates $S_{xx} \propto \mathbb{E}[\phi\phi^\top] = C$ and

$$S_{xy} \;\propto\; \mathbb{E}[\phi y] \;=\; (1-c)\,C\theta^* + c\,C(\theta^*+\Delta) \;=\; C(\theta^* + c\Delta), \tag{3}$$

so the normal equations give $\hat\theta = S_{xx}^{-1}S_{xy} = \theta^* + c\Delta$ exactly. The residual $y - \phi^\top\hat\theta$ has conditional mean $-c\,\phi^\top\Delta$ on normal windows (probability $1-c$). On anomalous windows (probability c) its

Algorithm 1 Dual-branch window update.

Require: window $X \in \mathbb{R}^{T\times n}$; causal graph A; invariant state $\{\theta_j, \sigma_j^2, S_{xx,j}, S_{xy,j}, S_{n,j}, S_{rr,j}\}$; hyperparameters $\pi, \lambda, \rho, \alpha, \kappa, \tau$; flag *trusted*
Ensure: anomaly score u; ranked root causes; updated invariant state
for each node j **do** ▷ variant refit: sparse mechanism shift; θ^{inv} held fixed
  $r_j \leftarrow y_j - \Phi_j\theta_j$ ▷ residual under the invariant branch
  $d_j \leftarrow \arg\min_d \|r_j - \Phi_j d\|^2 + \text{ridge}$ ▷ least squares on the residuals
  $\Delta_j \leftarrow \text{soft_threshold}(d_j,\ \lambda\sigma_j)$ ▷ prox step: sparse support
  $\tilde\sigma_j^2 \leftarrow \text{mean}\big((y_j - \Phi_j(\theta_j + \Delta_j))^2\big)$
end for
for each node j **do** ▷ model comparison: per-node gain
  $g_j \leftarrow (\ell_j^{\text{var}} - \ell_j^{\text{inv}}) - \frac{1}{2}(\|\Delta_j\|_0 + 1)\log T$
end for
$a \leftarrow \log\pi + \sum_j \ell_j^{\text{inv}}$; $b \leftarrow \log(1-\pi) + \sum_j \big(\ell_j^{\text{var}} - \frac{1}{2}\text{df}_j \log T\big)$
$w \leftarrow \text{softmax}(a, b)_1$ ▷ responsibility, equation 2
$u \leftarrow \big(b - \log(1-\pi) - \sum_j \ell_j^{\text{inv}}\big)/T$ ▷ per-sample penalized gain: detection
$w_{\text{used}} \leftarrow 1$ **if** *trusted* or routing $=$ `none` **else** w ▷ routing $\in \{$`em`, `residual`, `none`$\}$
if drift enabled **and not** *trusted* **then**
  $\text{streak} \leftarrow \text{streak} + 1$ **if** $u > \tau$ **else** 0 ▷ drift vs. anomaly; $\tau = \tau_\pi$ is $w_{\text{used}} < \frac{1}{2}$
  if $\text{streak} \geq \kappa$ **then**
    scale $(S_{xx}, S_{xy}, S_n, S_{rr})$ by α; $w_{\text{used}} \leftarrow 1$ ▷ absorb drift
  end if
end if
for each node j **do** ▷ gated invariant update
  $S_{xx,j} \leftarrow \rho S_{xx,j} + w_{\text{used}}\Phi_j^\top\Phi_j$; $S_{xy,j} \leftarrow \rho S_{xy,j} + w_{\text{used}}\Phi_j^\top y_j$
  $\theta_j \leftarrow (S_{xx,j} + \epsilon I)^{-1} S_{xy,j}$
  $S_{n,j} \leftarrow \rho S_{n,j} + w_{\text{used}}T$; $S_{rr,j} \leftarrow \rho S_{rr,j} + w_{\text{used}}\|y_j - \Phi_j\theta_j\|^2$
  $\sigma_j^2 \leftarrow S_{rr,j}/S_{n,j}$
end for
return u, $\text{rank_nodes}(\{g_j\})$, updated state

conditional mean is $(1-c)\,\phi^\top\Delta$. Its second moment is

$$\hat\sigma^2 = \sigma^2 + \big[(1-c)\,c^2 + c\,(1-c)^2\big]\,\mathbb{E}\big[(\phi^\top\Delta)^2\big] = \sigma^2 + c(1-c)\,q. \tag{4}$$

For the gated estimator, suppose the responsibilities separate window types, with $w \to 1$ on normal windows and $w \to 0$ on anomalous ones. The weighted statistics then converge to their clean-window restrictions. These restrictions recover θ^* and σ^2 for any $c < 1$. □

Proof of Proposition 2. On a given window the variant branch refits j^* on that window's data. Its population fit is therefore $(\theta^* + \Delta, \sigma^2)$ on anomalous windows. On normal windows the refit absorbs the reference's bias, so the fit is (θ^*, σ^2). The expected per-sample gain is the Gaussian Kullback–Leibler divergence from the variant fit to the polluted reference $(\hat\theta, \hat\sigma^2)$,

$$\text{KL} = \tfrac{1}{2}\Big[\log\tfrac{\hat\sigma^2}{\sigma^2} + \tfrac{\sigma^2+m^2}{\hat\sigma^2} - 1\Big], \qquad m^2 = \begin{cases}(1-c)^2\,q & \text{anomalous window,}\\ c^2\,q & \text{normal window,}\end{cases} \tag{5}$$

Here the mean discrepancy is $\phi^\top\big((\theta^*+\Delta) - (\theta^*+c\Delta)\big) = (1-c)\phi^\top\Delta$ on anomalous windows and $c\,\phi^\top\Delta$ on normal ones. Write $u = c(1-c)q/\sigma^2$ and expand equation 5 to first order in q/σ^2. The $\log(1+u)$ and $-u$ terms cancel, which leaves

$$\text{KL}_{\text{anom}} = \frac{(1-c)^2 q}{2\sigma^2} + O\Big(\tfrac{q^2}{\sigma^4}\Big), \qquad \text{KL}_{\text{norm}} = \frac{c^2 q}{2\sigma^2} + O\Big(\tfrac{q^2}{\sigma^4}\Big); \tag{6}$$

the exact expressions keep $\hat\sigma^2$ in the denominator, the variance-inflation deflation cited in the main text. At every unshifted node the variant refit fits only noise. Its expected penalized gain is therefore at most $-\frac{1}{2T}\,\text{df}\log T \leq 0$, with $O_p(T^{-1/2})$ fluctuations. Subtracting the two window types gives the pooled detection contrast $(1-2c)\,q/2\sigma^2 > 0$ for $c < \frac{1}{2}$. For localization on an anomalous window, g_{j^*}/T must exceed the largest penalized gain among the $n-1$ unshifted nodes. The maximum of $n-1$ centered $O_p(T^{-1/2})$ variables concentrates at $O\big(\sqrt{\log n/T}\big)$. Correct top-1 therefore requires $(1-c)^2 q/2\hat\sigma^2 \gtrsim \sqrt{\log n/T}$. This margin is multiplicative in q and $(1-c)^2$. □

Separation condition and scope. The gated claim needs the responsibility to separate window types. By equation 5 the per-window evidence gap scales as Tq/σ^2. Separation therefore holds once $Tq/\sigma^2 \gg \mathrm{df} \log T$. It fails in the weak-shift regime, where contaminating windows are themselves nearly clean. There gating has little effect in either direction. The detection-boundary sweep of Appendix C observes this condition empirically. The idealizations are a single shifted node, a well-specified linear-Gaussian mechanism, and steady state. The experiments of Section 5.2 relax all three. At finite T the variant support additionally spreads onto descendants. The theory omits this spread, which compounds the margin loss the propositions predict.

C Synthetic Protocol and Full Results

Stream construction. Streams come from linear or nonlinear SCMs with 12 nodes (edge probability 0.3) and Gaussian noise. An anomaly rescales one node's incoming weights or moves its noise mean, with strength s, and reaches downstream nodes only through propagation. Windows have $T = 200$ samples unless noted, 30 trusted warmup windows bootstrap the invariant branch, and results average 5 seeds. The heatmap sweep of Figure 2 uses 3 seeds with 200 windows per cell and paired window sequences across routing arms. inv-err is the relative Frobenius error of the invariant weights against ground truth. Detection AUC scores the penalized gain, and RCA top-1 is measured on truly anomalous windows.

Routing ablation. Table 6 compares the three routing arms of Algorithm 1. With strong, frequent anomalies the likelihood-ratio router dominates (RCA 0.949 vs. 0.531 without routing). It also keeps inv-err flat as contamination grows, and residual routing lets it rise. With weak shifts ($s = 0.3$) all three arms coincide, because the contaminating data is itself nearly clean. This is the boundary of the multiplicative law in Figure 2.

Table 6: Routing ablation on synthetic streams: likelihood-ratio (LR) routing vs. residual-magnitude routing vs. no routing, per regime (s, c, T). Bold: best per regime.

regime (s, c, T)	routing	inv-err	RCA top-1	AUC
1.0, 50%, 100	LR	**0.0150**	**0.949**	**0.999**
1.0, 50%, 100	residual	0.0209	0.906	**0.999**
1.0, 50%, 100	none	0.0220	0.531	0.992
1.0, 40%, 50	LR	**0.0184**	**0.913**	**0.995**
1.0, 40%, 50	residual	0.0206	0.900	0.994
0.3, 40%, 200	LR ≈ residual ≈ none	∼0.0105	0.93–0.95	∼0.965

Detection boundary. We sweep shift strength at 20% contamination. At $s = 0.15$ the detector reaches AUC 0.879 and RCA top-1 0.829. Both saturate (≥ 0.99 and ≥ 0.96) for $s \geq 0.5$. inv-err stays near 0.009 across the sweep, so gating holds at the detection boundary.

Drift vs. anomaly. We inject a permanent mechanism change at 2 nodes at window 150 of a stream with 15% transient contamination. Table 7 compares the adaptive persistence rule ($\kappa = 5$, model-implied τ_π) with a static reference. The static model alarms on every post-drift window (FPR 1.000). The adaptive rule re-converges within roughly 10 windows (FPR 0.058, inv-err 0.0067 against the drifted mechanism). Figure 4 shows the trajectory. Transient anomalies dip w for one window and leave the patience count at zero. At the drift point w stays pinned near 0 until absorption fires (same-seed post-drift FPR 0.074 adaptive vs. 1.000 static).

Table 7: Synthetic drift: permanent change at 2 nodes at window 150, contamination 15%, adaptive persistence rule against a static reference. Bold: best.

setting	FPR pre-drift	FPR post-drift	inv-err vs. drifted mechanism
adaptive ($\kappa = 5$)	0.000	**0.058**	**0.0067**
static	0.000	1.000	0.9560

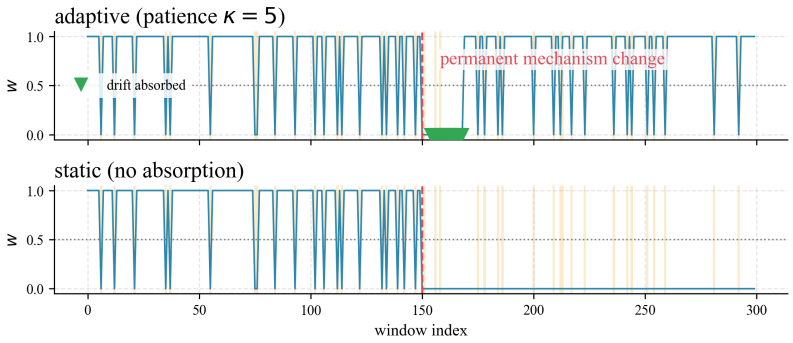

Figure 4: Responsibility w over a synthetic stream with transient anomalies (orange bands) and a permanent mechanism change at window 150 (red dashed line). Transient dips leave the patience count at zero. At the drift point w pins near zero, absorption fires (green markers), and the invariant branch re-converges within about ten windows. Without absorption (bottom) w stays pinned and the detector is permanently saturated.

Nonlinear mechanisms and misspecification. Table 8 reports tanh and sin SCMs at 20% contamination. With matching feature maps both reach AUC 1.000 and RCA 0.978. We then fit sin mechanisms with deliberately misspecified linear features. RCA remains 0.978 under likelihood-ratio gating against 0.676 without gating. The shift signal is a relative likelihood gain. Uniform misspecification therefore inflates the noise estimates of both branches, and the inflation largely cancels. The sampler benefit ($\approx +0.24$ RCA) replicates in all nonlinear settings.

Table 8: Nonlinear and misspecified settings at 20% contamination. Bold: best of each pair.

mechanisms	feature map	routing	AUC	RCA top-1
tanh	tanh (matched)	LR	1.000	0.978
sin	sin (matched)	LR	1.000	0.978
sin	linear (misspecified)	LR	n/a	**0.978**
sin	linear (misspecified)	none	n/a	0.676

D Real-Data Protocols and Full Results

D.1 PetShop: Protocol and Full Method Comparison

PetShop (Hardt et al., 2024) ships four traffic recordings (low, high, two temporal), 5-minute metrics, and 68 injected issues (36 latency, 32 availability). Each abnormal window holds only about 6 timestamps. The causal graph is the call-graph transpose, so edges run callee to caller. The protocol follows the official `rca_task.evaluate` harness: train plus test, top-k recall of the ground-truth node. Official baselines run in the benchmark authors' pinned environment (DoWhy 0.9.1 (Sharma & Kiciman, 2020), `sfr-pyrca` (Liu et al., 2023)).

Comparison with causal methods. Table 9 is the full comparison behind the PetShop entries of Table 2. Among causal methods the metric-aware dual-branch configuration is the strongest. It beats official CIRCA (Li et al., 2022) on top-1 (0.485 vs. 0.382) at equal top-3 (0.794). It also dominates RCD (Ikram et al., 2022) (0.265 / 0.309) and counterfactual attribution (Budhathoki et al., 2022) (0.176 / 0.426). It retrieves the ground truth in the top 3 for every latency issue.

Graph prior ablation. The upper block varies the upstream graph prior. The hard most-upstream prior is effective on sparse synthetic DAGs. On the dense real call graph it backfires (0.456 $\rightarrow$ 0.294 top-1). A single false-flagged upstream node hijacks the ranking. Softening the prior recovers latency (0.528 / 1.000) but hurts availability (0.406 $\rightarrow$ 0.250 top-1). Latency propagates along call edges and availability does not. Routing the prior per target metric, soft for latency and trigger-point for availability, recovers both (0.485 / 0.794) and is the reported configuration.

Associational baselines. Two engineered associational baselines, official ranked correlation and anomaly traversal, reach higher top-1 overall. Both gain from trigger-point search and tuned robust scorers, which are protocol advantages orthogonal to the mechanism model. We already adopt trigger-point selection. Rank-blending our gain with robust-z hurts (0.368 / 0.574), because the associational signal dilutes the mechanism attribution.

Table 9: PetShop, full method table: top-1 / top-3 recall over all 68 issues, with per-issue-type breakdowns. Upper block: our implementations under a shared protocol. Lower block: official implementations in the benchmark authors' environment. Bold: best per column within each block (ties bolded). †Default $\lambda = 0.10$, left untuned because PetShop has no held-out split (Table 15).

			latency		availability	
method	top-1	top-3	top-1	top-3	top-1	top-3
dual-branch (metric-aware)	**0.485**†	0.794	**0.528**	**1.000**	**0.438**	0.562
dual-branch (trigger, W=20)	**0.485**	0.735	0.500	0.889	**0.438**	0.562
dual-branch (soft + trigger)	0.397	**0.809**	**0.528**	**1.000**	0.250	0.594
dual-branch (soft upstream)	0.397	0.794	**0.528**	**1.000**	0.250	0.562
dual-branch (no upstream)	0.456	0.735	0.500	0.889	0.406	0.562
dual-branch (hard upstream)	0.294	0.426	0.250	0.361	0.344	0.500
dual-branch (trigger + z blend)	0.368	0.574	0.389	0.583	0.344	0.562
zscore (ours)	0.176	0.441	0.000	0.278	0.375	0.625
correlation (ours)	0.000	0.676	0.000	0.722	0.000	0.625
traversal (ours)	0.176	0.544	0.056	0.222	0.312	**0.906**
CIRCA (official)	0.382	0.794	0.500	**0.944**	0.250	0.625
RCD (official)	0.265	0.309	0.111	0.167	0.438	0.469
anomaly traversal (official)	0.500	**0.868**	**0.667**	0.750	0.312	**1.000**
ranked correlation (official)	**0.647**	0.779	0.556	0.694	**0.750**	0.875
ϵ-diagnosis (official)	0.015	0.015	0.028	0.028	0.000	0.000
counterfactual attribution (official)	0.176	0.426	0.333	0.556	0.000	0.281

D.2 RCAEval RE1: Protocol and Full Results

RCAEval RE1 (Pham et al., 2025; 2024b) contains 375 failure cases from Online Boutique (OB), Sock Shop (SS), and Train Ticket (TT). Each system contributes 5 fault types (CPU / MEM / DISK / DELAY / LOSS) × 5 services × 5 repetitions. The telemetry is metric-only at roughly 1 s sampling. The official protocol hands every method the fault-injection timestamp ($t_\Delta = 0$).

Metric set. SS and TT ship two files per case. `simple_data.csv` has one column per service × {cpu, mem, latency, ...} (59 and 220 columns). `data.csv` is the raw container-level export (437 and 1241 columns). RCAEval's documented metric counts (49–212) and BARO's published accuracy correspond to the simple file, which is the official protocol and the one used here. OB ships only `data.csv`, already in simple form (50 columns), so its numbers match in both regimes. Section 5.3 studies the raw regime deliberately.

Splits and causal graph. All methods share the identical split: at most 1800 normal samples before the injection and 120 samples after. RE1 lacks a call graph, so the causal graph is estimated from the normal period. Each metric receives its top-3 $|\mathrm{corr}|$ parents, which exercises the data-driven graph path. The service score is the max node gain over the service's metrics. CIRCA/HT and RCD are the `sfr-pyrca` implementations (Liu et al., 2023). BARO is the official `fse-baro` package (Pham et al., 2024a), which segments with multivariate BOCPD (Adams & MacKay, 2007). The nsigma baseline is a robust-z scorer in the family of ϵ-diagnosis (Shan et al., 2019).

Tables 10 and 11 give the full results. BARO wins this benchmark clearly and leads on every system and every fault type. Among causal methods we trail CIRCA on AC@1 (0.347 vs. 0.533) and are level at AC@3 (0.691 vs. 0.696). We are close at Avg@5 (0.645 vs. 0.688) and clearly beat RCD (0.237 / 0.301). Our method is strongest on MEM (0.792, second only to BARO) and CPU (0.747), which are node-local mechanism changes. It is weakest on DELAY (0.584 vs. BARO's 0.829),

the most propagation-heavy fault type, and on TT (0.386), the largest system. Appendix E gives significance tests for these comparisons.

Table 10: RCAEval RE1 under the official metric set (375 cases). Bold: best per column.

method	AC@1	AC@3	Avg@5	OB Avg@5	SS Avg@5	TT Avg@5
BARO (official pkg)	**0.685**	**0.880**	**0.844**	**0.878**	**0.955**	**0.698**
CIRCA / HT (pyrca)	0.533	0.696	0.688	0.782	0.814	0.467
dual-branch (ours)	0.347	0.691	0.645	0.742	0.808	0.386
RCD (pyrca)	0.237	0.301	0.301	0.550	0.275	0.077
nsigma / robust-z	0.104	0.448	0.415	0.528	0.611	0.106

Table 11: RE1 breakdowns: AC@1 by system, Avg@5 by fault type. Bold: best per column.

	AC@1 by system			Avg@5 by fault type				
method	OB	SS	TT	CPU	MEM	DISK	DELAY	LOSS
BARO	**0.720**	**0.840**	**0.496**	**0.907**	**0.984**	**0.907**	**0.829**	**0.592**
CIRCA / HT	0.616	0.672	0.312	0.757	0.779	0.579	0.752	0.573
dual-branch	0.456	0.464	0.120	0.747	0.792	0.568	0.584	0.536
RCD	0.480	0.184	0.048	0.413	0.296	0.440	0.117	0.237
nsigma	0.064	0.184	0.064	0.432	0.528	0.341	0.413	0.360

Efficiency. Table 12 reports median seconds per case on an otherwise idle Apple M4 Pro laptop (Appendix G). Efficiency does not separate us from BARO (0.02–0.06 s vs. our 0.01–0.06 s). The efficiency argument holds only against RCD. RCD is roughly $180\times$ more expensive than either and the least accurate causal method. CIRCA sits in between (roughly 2–3$\times$).

Table 12: Median runtime per RE1 case in seconds.

method	OB (47 metrics)	SS (52)	TT (211)
nsigma	0.00	0.01	0.02
BARO	0.02	0.02	0.06
dual-branch	0.03	0.01	0.06
CIRCA / HT	0.03	0.04	0.17
RCD	4.77	1.72	8.64

D.3 Service-Level Aggregation Ablation

This ablation measures how much of the RE1 gap comes from pooling per-metric gains into one service score. Every arm in Table 13 uses the same per-metric gains, and only the metric-to-service reduction changes. `fam_z` standardizes gains within each metric family (cpu, mem, latency percentiles, ...) before taking the max, so a family with globally larger gains cannot dominate. `oracle_fam` restricts the max to the families the injected fault should move. It reads the fault label, so it is only an upper bound.

Aggregation explains part of the AC@1 gap. A label-free change of reduction buys $+5.8$pp AC@1 ($0.347 \rightarrow 0.405$) and $+4.2$pp AC@3 at zero modeling cost. Even the family-selection oracle reaches only 0.520, still 16.5pp below BARO's 0.685. Better pooling cannot close the gap. The remaining deficit lives in the per-metric gains themselves.

`count_pos` is much worse than every continuous reduction (0.229). Counting BIC-significant metrics throws away effect size. On a correlation-estimated graph, that count mostly reflects how many parents a metric happened to receive, and only weakly reflects fault severity.

Table 14 shows the split by fault type. `fam_z` sharply helps node-local resource faults (DISK $0.173 \rightarrow 0.493$, CPU $+14.6$pp, MEM $+9.3$pp) and sharply hurts propagating network faults (DELAY $0.240 \rightarrow 0.080$). The reason is structural. A network fault shifts an entire latency family at

once, and standardizing within that family subtracts exactly the common shift that identifies it. We report `max` in Table 10 to keep that table a clean single-configuration comparison. We treat `fam_z` as the recommended default when resource faults dominate.

Table 13: RE1 aggregation ablation: same per-metric gains, different service reductions. The oracle row (italic) uses the fault label and is an upper bound. Bold: best label-free rule.

aggregation	AC@1	AC@3	Avg@5
`oracle_fam` (upper bound, uses fault label)	*0.520*	*0.824*	*0.771*
`fam_z` (within-family standardization)	**0.405**	**0.733**	**0.681**
`sum_pos` (sum of significant gains)	0.363	0.728	0.669
`top2` (mean of two largest)	0.357	0.720	0.663
`mean`	0.347	0.709	0.650
`max` (default in Table 10)	0.347	0.691	0.645
`count_pos` (number of significant metrics)	0.229	0.528	0.504

Table 14: AC@1 of `fam_z` vs. `max` by fault type: within-family standardization helps node-local faults and hurts propagating ones. Bold: best per column.

aggregation	CPU	MEM	DISK	DELAY	LOSS
`max`	0.507	0.560	0.173	**0.240**	**0.253**
`fam_z`	**0.653**	**0.653**	**0.493**	0.080	0.147

D.4 PetShop Hyperparameter Sensitivity

The full grid is $\lambda \in \{0.02, 0.05, 0.10, 0.20, 0.40\} \times W \in \{20, 40, 80, 160\}$ for both reported configurations, 68 issues each. Table 15 fixes λ at its default 0.10 and varies W. Accuracy changes mildly and monotonically with W, with no cliff. The metric-aware configuration is the more stable of the two. Its top-3 is 0.794 at all 20 grid points, and its latency top-3 is 1.000 throughout.

The λ dependence is diagnostic. For the trigger configuration all five λ values give identical top-1 and top-3 at every W. PetShop's injected shifts are tens of σ after standardization, so the ordering of raw gains is invariant to the threshold. For the metric-aware configuration top-1 moves from 0.500 at $\lambda \leq 0.05$ to 0.485 at $\lambda \geq 0.10$ for $W \in \{20, 40\}$. The spread is 1.5pp there and rises to 2.9pp at $W = 160$, while top-3 stays at 0.794. The soft upstream prior depends on which nodes are flagged ($g_j > 0$), and a different λ changes the support of Δ and hence df_j. For borderline nodes this flips the sign of the penalized gain and changes the flagged set. The sparsity prior is therefore inert where effect size alone drives the ranking and live once a discrete decision is layered on it. We keep $\lambda = 0.10$ for the headline numbers even though 0.02–0.05 would give 0.500. PetShop has no held-out split, so selecting the threshold on these 68 issues would be tuning on the test set.

Table 15: PetShop sensitivity to window length W at the default $\lambda = 0.10$, for both reported configurations (68 issues each).

	trigger		metric-aware	
W	top-1	top-3	top-1	top-3
20	0.485	0.735	0.485	0.794
40	0.471	0.735	0.485	0.794
80	0.456	0.721	0.456	0.794
160	0.426	0.750	0.441	0.794

D.5 Detection, Autonomous Operation, and Drift on Real Telemetry

Detection protocol. Both benchmarks annotate the injection time, so the detection read-off of Section 5.2 can be scored on real telemetry. BARO, CIRCA, and RCD take the anomaly timestamp as an

input, so detection lies outside their interface. We compare against five detectors behind one streaming interface. Robust-z takes $\max|z|$ over the window with median and MAD from the warmup. BOCPD (Adams & MacKay, 2007) is scored by Bayesian surprise, the peak negative log one-step predictive evidence within the window. ADWIN (Bifet & Gavaldà, 2007) and Page–Hinkley (Page, 1954) come from `river` 0.21 (Montiel et al., 2021), and the windowed kernel change statistic from `ruptures` 1.1.10 (Truong et al., 2020). The kernel statistic is the per-metric one-segment minus two-segment CostL2 at the boundary with the previous window. Every detector receives the identical warmup slice and window sequence as GDRC, runs per metric where univariate, and aggregates by max. Page–Hinkley's firing threshold is calibrated on the warmup. Each model is warmed on an early normal slice and then scores the remaining normal windows (negatives) and the injected window (positive) in streaming order. Table 16 reports pooled ROC AUC and AP. AP is the headline metric because positives are 0.9% (PetShop) and 13% (RE1) of windows.

Three implementation choices keep the baselines from becoming strawmen. First, under a constant hazard H the BOCPD recursion normalizes the change-point mass $P(r_t{=}0)$ to exactly H at every step. That mass is therefore useless as a score. Second, surprise is taken as the peak within a window. BOCPD adapts within a few steps of a level shift, so the window mean would average the shift away. Third, a multivariate RBF kernel cost misses one shifted metric among twenty, and the per-metric CostL2 contrast finds it. Synthetic streams with one shifted metric in twenty confirm this. All five baselines separate at 4σ, three at 2σ, and the kernel statistic and Page–Hinkley at 1σ.

Table 16: Detection on real telemetry: pooled ROC AUC and average precision of the penalized-gain score against five change detectors given identical warmup and windows. PetShop AP is also split by target metric. Bold: best per column.

	PetShop ($68/7444$, prevalence 0.9%)				RE1 ($375/2500$, 13.0%)	
detector	AUC	AP	AP avail.	AP latency	AUC	AP
GDRC (ours)	**0.990**	**0.601**	0.863	**0.729**	0.837	0.620
robust-z	0.803	0.575	**0.934**	0.241	0.746	0.558
kernel change (ruptures)	0.980	0.535	0.887	0.333	**0.973**	**0.863**
Page–Hinkley	0.841	0.504	0.786	0.271	0.863	0.524
BOCPD	0.974	0.182	0.206	0.166	0.790	0.264
ADWIN	0.472	0.009	0.009	0.009	0.623	0.235
random	0.500	0.009	0.009	0.009	0.500	0.130

By traffic scenario, GDRC leads in three of the four PetShop scenarios (AP 0.725, 0.602, 0.599). In low traffic it trails the kernel statistic by 0.013 (0.721 vs. 0.734). On RE1 the kernel statistic leads on every system (OB 0.787, SS 0.997, TT 0.954 against our 0.431, 0.734, 0.747). It also leads on four of the five fault types, with a tie on MEM (0.713 vs. 0.716). ADWIN is near chance on PetShop because it adapts its reference right after the transition, so a single post-injection window often stays silent. That loss belongs to the evaluation design.

Autonomous operation protocol. Each case becomes one continuous stream. It opens with a trusted warmup of 35% of the pre-injection windows. The first half of the warmup trains the invariant branch and the second half calibrates the alarm threshold. The remaining healthy windows follow, then the injection and its aftermath. Labels and the injection time stay hidden from every detector. The alarm threshold is $\mathrm{thr} = \max(c) + (m-1)\,(\max(c) - \mathrm{median}(c))$ over the calibration scores c. It is swept over margins $m \in \{1, 1.5, 2, 3, 5, 8, 12, 20, 35, 60, 100\}$ and applied identically to every detector. The rule is additive because the penalized gain is negative on well-fit healthy windows. Calibrating on the whole warmup is a trap. The first warmup window is scored against an untrained invariant branch, so its gain is enormous. In our first run that single value made every alarm unreachable (recall 0.000 at every margin). Windows are $W = 5$ (PetShop, 5-minute samples) and $W = 20$ (RE1, 1-second samples). The 120-sample RE1 abnormal stretch therefore yields six windows, so delay has resolution. Table 17 reports recall and false alarms per hour at four margins. RE1 has 180 windows per hour, so one false alarm per hour is 0.6% of windows.

For GDRC the ranking is read at three references on the same detected cases. The first is the window of our own alarm, which is fully autonomous. The second is the true injection window with the same window length. The third is the whole abnormal stretch scored as one window, the offline protocol.

Table 17: End-to-end streaming detection: recall / false alarms per hour at four threshold margins m. Every detector sees only the stream, with labels and injection times withheld. On PetShop the gain leads at every budget below one false alarm per hour. On RE1 the kernel statistic leads.

	PetShop (68 streams, 5-min windows)				RE1 (375 streams, 20-s windows)			
detector	m=1	m=3	m=12	m=100	m=1	m=3	m=12	m=100
GDRC (ours)	0.76 / 0.26	0.66 / 0.14	0.54 / 0.07	0.43 / 0.06	0.90 / 69.8	0.85 / 53.2	0.78 / 33.9	0.66 / 18.0
robust-z	0.65 / 0.47	0.44 / 0.06	0.13 / 0.00	0.00 / 0.00	0.85 / 37.6	0.71 / 23.0	0.55 / 15.0	0.25 / 9.5
kernel change (ruptures)	0.71 / 0.54	0.65 / 0.19	0.44 / 0.17	0.43 / 0.08	1.00 / 31.1	0.98 / 15.2	0.92 / 5.3	0.71 / 1.3
Page–Hinkley	0.79 / 2.08	0.66 / 0.99	0.31 / 0.10	0.29 / 0.10	0.87 / 43.6	0.44 / 17.5	0.16 / 11.4	0.14 / 11.4
BOCPD	0.35 / 0.54	0.35 / 0.23	0.31 / 0.08	0.12 / 0.02	0.62 / 31.2	0.54 / 19.0	0.42 / 9.3	0.19 / 2.9
ADWIN	0.00 / 0.27	0.00 / 0.14	0.00 / 0.11	0.00 / 0.11	0.57 / 16.0	0.18 / 3.8	0.03 / 0.8	0.02 / 0.7

The first gap isolates the cost of alarming late and the second the cost of short windows. All three are averaged over exactly the detected subset. Comparing a conditional metric with an oracle over all cases would be a selection effect. Table 3 in the main text gives the decomposition at margin 1. On PetShop an abnormal stretch is a single window, so the three references coincide and the row is listed for completeness. The offline RE1 AC@1 of 0.347 exceeds every streaming reference because of the reference budget. The offline protocol trains on up to 1,800 trusted samples in 200-sample windows and estimates the correlation graph from them. The median stream trains on two trusted 20-sample windows, and later healthy windows enter only through the gate.

Drift selectivity protocol. This experiment asks whether the method can tell a real traffic change from a fault. PetShop ships four traffic regimes (low, high, temporal1, temporal2), each with an issue-free recording and injected issues. For each ordered regime pair (S, S') and each issue in S' we build the stream $\mathrm{warmup}(S) \to \mathrm{healthy}(S) \to \mathrm{healthy}(S') \to \mathrm{fault}(S')$. It has 12 healthy, 12 post-switch, and one fault window of five samples. Each stream is standardized by the regime it warms up in, which is all an online monitor has at the start. Alignment leaves 96 availability and 108 latency cases over all twelve ordered pairs. Three rules run on every stream. The static reference never absorbs. The model-implied threshold τ_π absorbs after κ windows with $w < \frac{1}{2}$. The calibrated threshold absorbs after $\kappa = 3$ consecutive windows above the warmup-calibrated alarm threshold (margin 3). On absorption the invariant statistics decay by $\alpha = 0.5$ and the window enters at full weight, so an absorbed fault is a missed fault. Table 4 in the main text reports the post-switch alarm and absorption fractions and the detection and absorption rates of the following fault.

The static reference alarms on 71% of availability and 45% of latency post-switch windows. The calibrated rule halves this. The residual is structural, because $\kappa = 3$ of the 12 post-switch windows alarm before absorption can fire. That gives a floor of 0.25. The model-implied rule absorbs in every case, as the saturation of w predicts (Section 5.3). On latency this even raises fault detection, because a one-window fault against a reference refreshed every window is sharp. On availability it absorbs 12.5% of the faults. Table 18 restricts to the cases in which the calibrated rule absorbed the regime change and compares it with the static reference on the identical stream. Drift alarms drop from every window to 0.39 and 0.28. Detection of the subsequent fault is unchanged (zero discordant availability cases, one latency case).

Table 18: Paired selectivity test on the cases in which the calibrated rule absorbed the regime change: static vs. calibrated reference on the identical stream. Absorbing the benign change removes most drift alarms and leaves detection of the subsequent fault unchanged.

		drift FA / window		fault detected		
metric	n	static	calibrated	static	calibrated	discordant
availability	56	1.000	0.393	0.929	0.929	0
latency	42	1.000	0.278	0.810	0.786	1

Two caveats apply. A fault-only control arm ($\mathrm{warmup}(S') \to \mathrm{healthy}(S') \to \mathrm{fault}(S')$) is archived and is an imperfect control. It standardizes the fault by the target regime, while the drifted arm standardizes by the source. The drifted arm therefore sees a larger raw deviation (latency fault detection 0.611 vs. 0.824 under the static rule). PetShop availability is near constant while healthy. The invariant noise variance therefore reaches its floor and per-node gains tie in groups. The archived

RCA columns use pessimistic tie-breaking and report the mean number of tied nodes (0.48 for availability, 0 for latency).

Regime concatenation. Table 19 reports the earlier concatenation test (latency metrics, $W = 20$). Each stream is one scenario's issue-free period followed by another's, so every post-switch alarm is false. This run used the model-implied rule with $\kappa = 5$, which fires every window on PetShop latency (see below), so the test measures adaptation only. The adaptive model brings the post-switch false-alarm rate from 1.000 to 0.069, close to the synthetic $1.000 \to 0.058$. Between two nominally equivalent temporal scenarios both settings stay near silent (0.02–0.07), so the rule responds to the regime change itself.

Table 19: Regime concatenation on real telemetry: false-alarm rates before and after concatenating two issue-free PetShop traffic scenarios under the model-implied rule. The temporal pair is a negative control. Bold: the regime-change contrast.

stream	rule	pre-switch	post-switch
low → high traffic, static	calibrated alarm	0.000	**1.000**
low → high traffic, adaptive	calibrated alarm	0.000	**0.069**
temporal1 → temporal2, static	calibrated alarm	0.037	0.024
temporal1 → temporal2, adaptive	calibrated alarm	0.074	0.061

Saturation of the responsibility. The responsibility w is essentially 0 on healthy PetShop windows at every window length from 5 to 120 and even on 4-node subsets. The cause is misspecification. PetShop latency is non-stationary and heavy-tailed within the issue-free period, so refitting per window always pays. The likelihood ratio then always favors the variant branch. The ranking of the score still transfers to real data (AUC 0.990 / 0.837), but the absolute responsibility stays pinned near zero. Both the alarm threshold and the drift threshold τ are therefore calibrated on the warmup on real telemetry (Section 4.3). The model-implied τ_π appears above only as an ablation.

E SIGNIFICANCE TESTING

Methodology. Every benchmark number in the paper is a point estimate over a fixed case set. Both benchmarks are paired, because every method scores the same failure case. We therefore test AC@k with McNemar's exact test on the discordant pairs. We test Avg@5 with a paired bootstrap (10,000 resamples, fixed seed). The script first verifies pairing by checking that the (system, fault) sequence is identical across methods. It reports a result only when this check passes. The official-harness PetShop baselines lack per-issue outcomes, so only same-harness comparisons are testable.

Table 20 gives the full RE1 comparisons ($n = 375$ paired cases, curated metric set). Our AC@3 is statistically indistinguishable from CIRCA's, with 49 cases won only by CIRCA and 47 only by us ($p = 0.92$). "Level at top-3" is therefore a tested statement. Every other RE1 comparison is significant at $p < 0.001$, including BARO's lead over us on both metrics. The paired bootstrap puts CIRCA's Avg@5 advantage at $+0.043$ with CI $[+0.003, +0.082]$, so CIRCA is significantly ahead on Avg@5 even though the margin is small.

Table 20: RE1 significance tests, curated metric set, $n = 375$ paired cases. Diffs are (first method − second). b/c counts cases won only by the first method and only by the second. p is McNemar exact. ***: $p < 0.001$. n.s.: not significant. Bold: the single non-significant comparison.

comparison	AC@1 diff	b/c	p	AC@3 diff	b/c	p
BARO vs. ours	+0.339	152/25	2e−23 ***	+0.189	89/18	2e−12 ***
BARO vs. CIRCA	+0.152	95/38	8e−07 ***	+0.184	92/23	6e−11 ***
CIRCA vs. ours	+0.187	94/24	6e−11 ***	**+0.005**	**49/47**	**0.92 n.s.**
ours vs. RCD	+0.109	89/48	6e−04 ***	+0.389	168/22	5e−29 ***
ours vs. nsigma	+0.243	103/12	3e−19 ***	+0.243	107/16	1e−17 ***

PetShop configuration tests. On PetShop ($n = 68$) the ordering among our own configurations is only partly resolvable. Metric-aware beats the hard upstream rule decisively (top-1 $+0.191$, $p = 0.004$, and top-3 $+0.368$, $p = 4\text{e}{-07}$). The test cannot separate it from the simpler no-upstream variant (top-1 $+0.029$, $p = 0.82$, and top-3 $+0.059$, $p = 0.13$ with $4/0$ discordance). We therefore recommend metric-aware on mechanistic grounds and for its window stability (Table 15). Its gain over no-upstream remains undemonstrated, because 68 issues cannot separate the two.

F Edge-Level Δ Attribution: A Negative Result

Network faults (DELAY / LOSS) inject latency or packet loss on the communication between two services. The natural hypothesis is therefore that the shift lives on a graph edge. We tested it with an edge-level readout. Each parent-child pair $(i \to j)$ is scored by its normalized coefficient shift $|\Delta_{ij}|$ and attributed to the parent's service. The model and fits are identical to Table 10, and only the ranking changes. Tables 21 and 22 report the official metric set.

Table 21: Edge-level readout on RE1, official metric set. Bold: best per column.

readout	AC@1	AC@3	Avg@5
node gain (default)	**0.347**	**0.691**	**0.645**
edge-Δ magnitude only	0.032	0.131	0.123
min-rank fusion (node, edge)	**0.347**	0.600	0.532

Table 22: Avg@5 on network faults (DELAY + LOSS) by system. The node readout wins on every system, even on the faults the edge hypothesis targets. Bold: best per column.

readout	OB	SS	TT
node gain	**0.54**	**0.76**	**0.38**
edge only	0.28	0.21	0.06
min-rank fusion	0.45	0.63	0.27

The tables support three findings. First, edge-level attribution is far weaker than node-level attribution everywhere (Avg@5 0.123 vs. 0.645). It cannot replace the node readout. Second, fusion fails to rescue it. Min-rank fusion trails node gain on every system, including TT network faults (0.27 vs. 0.38). An earlier raw-metric run showed a marginal fusion gain on TT network faults (0.224 vs. 0.216). It vanishes under the corrected protocol, so we withdraw it. Third, the diagnosis is graph quality, and the idea stays intact. RE1 ships no call graph, so parents are top-3 correlation neighbors. A correlation edge between two metrics differs from a communication channel. $|\Delta_{ij}|$ therefore measures "which co-movement broke," which only coincidentally tracks the faulty link.

We keep node gain as the default. Edge-level Δ becomes an extension gated on trace-derived topology, where $|\Delta_{ij}|$ would have a physical interpretation, as in RCAEval RE2/RE3. The negative result delimits the scope of the sparse-mechanism-shift assumption. It holds for node mechanisms, and metric-only telemetry does not expose the edge mechanisms that network faults perturb.

G Reproducibility Details

Script-to-result mapping. Every number in the paper is produced by a script in the supplementary material. The mapping is as follows.

- `experiment.py`: contamination robustness (Table 1).
- `experiments.py`: detection boundary, synthetic drift, and nonlinear results (Tables 7 and 8 in Appendix C).
- `experiments_hard.py`: the weak-shift and short-window regimes of Table 6.
- `petshop_experiment.py`: Table 9. With `--sens`: Table 15.

- `rcaeval_experiment.py`: Tables 10 and 11 (dual-branch, nsigma, BARO). With `--agg`: Tables 13 and 14. With `--edge`: Tables 21 and 22. With `--full`: the raw-telemetry arm of Section 5.3.
- `rcaeval_baselines.py` (`circa` or `rcd`, optional `--full`): the causal baselines of Table 10 and of the raw arm.
- `timing_experiment.py`: Table 12.
- `rate_ablation.py`: the counter-differencing control of Section 5.3 (`rcaeval_results_rate.csv`).
- `family_ablation.py`: the column-family redundancy tiers of Section 5.3 (`rcaeval_results_families.csv`).
- `significance.py`: Appendix E (`results/significance.txt`).
- `skeleton_ablation.py`: the causal-graph ablation of Section 5.3 (`results/petshop_skeleton.csv`).
- `real_detection_drift.py` (detectors in `detection_baselines.py`): Tables 16 and 19 (`results/real_detection.csv`, `results/real_drift.csv`).
- `streaming_eval.py`: Tables 3 and 17 (`results/streaming_eval.csv`).
- `drift_selectivity.py`: Tables 4 and 18 and the drift column of Table 2 (`results/drift_selectivity.csv`).
- `archive_results.py`: re-runs every synthetic and PetShop table and writes the `results/` archive. `make_figures.py` produces Figures 2 and 4. `make_method_figure.py` produces Figure 1. Figure 3 is drawn from the archived aggregation, causal-graph, and encoding CSVs.

Result-file inventory. The top level holds `rcaeval_results*.csv` (official metric set), `_full.csv` (raw metric set), and `timing_results.csv`. The `results/` directory holds the per-run archives. For the synthetic sections these are `synth_contamination.csv`, `synth_magnitude.csv`, `synth_drift.csv`, `synth_sampler.csv`, and `synth_nonlinear.csv`. Each has one row per seed, so dispersion can be recomputed. For PetShop they are `petshop_methods.csv` (one row per issue $\times$ method) and `petshop_sensitivity.csv` (the full $\lambda \times W$ grid). Every table in the paper is derivable from these files, which are included in the supplementary material.

Hardware. All experiments, including the runtimes of Table 12, ran CPU-only in one process on an Apple M4 Pro laptop (14-core CPU, 24 GB unified memory, macOS 26.5). Every method, including the baselines, is classical and non-neural.

Software and pinned environments. Our method and the synthetic experiments depend only on NumPy and pandas (Matplotlib for figures) and run unchanged under Python 3.10–3.13. The archived result files were produced inside the pinned `petshop-bench` conda environment (Python 3.10.20, NumPy 1.23.2, pandas 1.5.3). This environment is the PetShop benchmark authors' own. It also provides DoWhy 0.9.1 (Sharma & Kiciman, 2020), `sfr-pyrca` 1.0.1 (Liu et al., 2023), scikit-learn 1.1.3, statsmodels 0.13.5, and NetworkX 2.8.8. The `sfr-pyrca` package supplies the CIRCA/HT and RCD implementations used on RE1. BARO is the official `fse-baro` 0.2.2 package (Pham et al., 2024a). The change-detection baselines use `ruptures` 1.1.10 and `river` 0.21.0. Installing `river` upgrades pandas, which must then be pinned back to 1.5.3 for the `sfr-pyrca` baselines. Synthetic experiments use fixed RNG seeds (5 seeds). The exact commands, seeds, and environment files are in the README of the supplementary material.

Datasets. PetShop (Hardt et al., 2024) is distributed under CC-BY-4.0. The RE1 split of RCAEval (Pham et al., 2025) is available from Zenodo under DOI 10.5281/zenodo.14590730.

Reporting requirement. Section 5.3 shows that RE1 rankings swing by tens of points with the input encoding. Any RE1 number should therefore state the metric file (`simple_data.csv` vs. `data.csv`) and the counter handling (cumulative vs. differenced). We release all three arms (curated, raw, raw-differenced) in the supplementary material, so both choices are auditable.