

Original Research

Interpretable Anomaly Explanation in Distributed Traces Using Causal Graphs and Counterfactual Ranking

Phạm Ngọc Hải¹ and Nguyễn Bảo Khánh²

¹Da Nang Institute of Technology and Management, Management and Computer Science Department, Nguyễn Văn Linh Street, Đà Nẵng 550000, Việt Nam.

²Southern Mekong University of Applied Sciences, Management and Computer Science Department, Trường Chinh Road, Cần Thơ 900000, Việt Nam.

Abstract

Modern microservice platforms increasingly rely on distributed tracing to observe end-to-end behavior across heterogeneous components, runtimes, and networks. While anomaly detection over traces has matured, operators still face a gap between detecting that a trace is abnormal and understanding why it is abnormal in a way that is both actionable and stable under noise, sampling, and instrumentation drift. This paper develops a technically grounded approach for interpretable anomaly explanation in distributed traces by combining causal graphs with counterfactual ranking. The central idea is to lift raw trace events into a causal representation that distinguishes correlation from plausible influence paths, then to generate explanation candidates as minimal interventions whose predicted counterfactual effect reduces abnormality. The method unifies structural causal models over trace-derived variables, graph-constrained representation learning for node and edge attributes, and a ranking objective that trades off explanatory power, intervention cost, and robustness. The approach supports backprop-friendly scoring for learning components, while retaining explicit causal semantics for explanation. It further addresses distributed execution mechanics, storage internals, and complexity constraints typical of high-cardinality trace streams. Error analysis is provided for approximate query execution and sketch-based feature aggregation, and multi-objective optimization is used to navigate Pareto trade-offs among latency overhead, explanation fidelity, and operational cost. Evaluation methodology is specified to enable reproducible benchmarking across services, workloads, and failure modes.

1. Introduction

Distributed traces encode a temporally ordered set of spans that represent causally linked work across services [1]. Each span contains timestamps, parent-child relationships, tags, logs, and resource attributes, and traces compose into graphs that approximate request execution. In production, traces are high-cardinality, partially sampled, and instrumented unevenly. Detection pipelines typically compress this complexity into scalar anomaly scores over trace-level or service-level aggregates [2]. The operational difficulty begins after an alert fires: an operator needs an explanation that identifies what changed, where it changed, and which changes are likely to matter. A useful explanation must be interpretable, meaning it can be expressed as a small set of variables and relations; causal, meaning it distinguishes plausible influence from spurious correlation induced by confounding, selection, or shared load; counterfactual, meaning it supports reasoning of the form “if this component had not regressed, would the trace still be anomalous”; and rankable, meaning multiple candidates can be prioritized under constraints like effort, risk, and blast radius.

Explanations for trace anomalies are difficult because traces are not i.i.d. data points [?]. A trace is a structured object with graph topology, time, and resource context, and anomalies can arise from path changes, fan-out amplification, queuing, retries, or partial outages. Moreover, observed trace graphs are imperfect proxies for underlying causal processes. Missing spans, skewed clocks, adaptive sampling, and

dynamic routing all introduce systematic biases. A naive correlation-based approach often over-explains: it highlights many correlated symptoms without isolating causes [3]. Conversely, a purely mechanistic approach that assumes complete knowledge of dependencies under-explains when the system is evolving or partially observed.

This paper presents a framework that constructs a causal graph from trace data and then generates explanations via counterfactual ranking. The framework treats trace-derived variables as nodes in a structural causal model constrained by graph topology and temporal ordering. An anomaly score is defined as a differentiable function over a trace representation that combines span-level metrics and learned embeddings [4]. Explanation candidates are interventions on a small subset of variables, constrained by causal structure and operational feasibility. Candidates are evaluated by counterfactual inference: the predicted anomaly score under intervention is compared with the observed anomaly score, producing an explanatory effect. Candidates are then ranked by a multi-objective criterion that balances effect size, intervention cost, robustness to uncertainty, and computational overhead.

A key theme is the tension between interpretability and scalability [5]. Tracing backends operate at extremely high ingest rates; explanation must be computed within latency and budget constraints, often in near real time. The framework therefore includes performance engineering elements: compact feature vectors, approximate aggregation via sketches, low-rank factorization of service interaction matrices, and indexing strategies for fast candidate generation. It also includes rigorous error analysis to quantify how approximation affects counterfactual ranking. Throughout, the goal is not to claim perfect causality from imperfect observability, but to produce explanations that are more stable and operationally aligned than correlation-only baselines, while being implementable in distributed trace analytics systems [6].

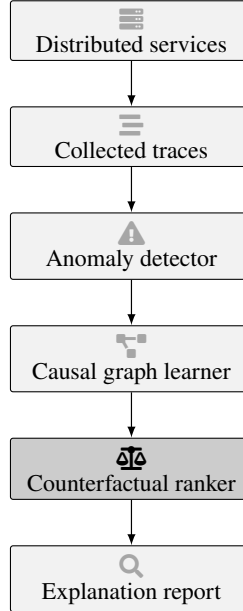


Figure 1: System-level view of the anomaly explanation pipeline built on distributed traces, causal modeling, and counterfactual ranking.

2. Problem Setting and Formalization

A distributed trace is modeled as a directed acyclic graph whose vertices are spans. Each span s has attributes including service identifier $\sigma(s)$, operation name, start time t_s^{start} , end time t_s^{end} , duration $d_s =$

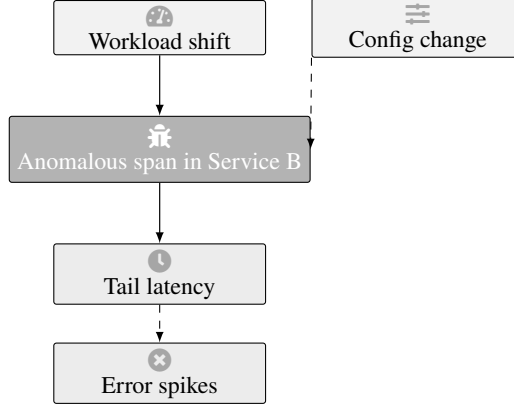


Figure 2: Local causal neighborhood around an anomalous span: upstream factors such as workload and configuration influence the span, which in turn drives observable symptoms in the trace.

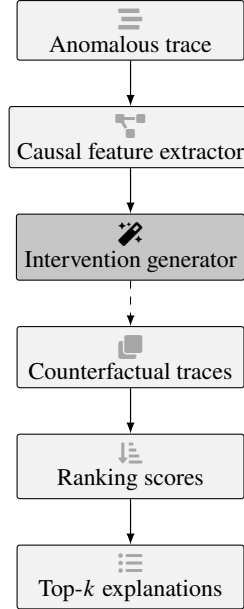


Figure 3: Counterfactual ranking: from an anomalous trace, causal features drive interventions that synthesize counterfactual traces, whose impact on anomaly scores yields an ordered list of explanations.

$t_s^{\text{end}} - t_s^{\text{start}}$, status code, tags, and resource metrics sampled during execution. Parent-child edges represent synchronous call structure or inferred causality via context propagation. In practice, some edges are missing and some are inferred, but the observed trace graph is a natural scaffold for causal constraints.

Let $G_T = (V_T, E_T)$ denote the observed graph of a trace T [7]. For each span $s \in V_T$, define a feature vector $x_s \in \mathbb{R}^p$ consisting of normalized duration, queue time estimates, retry counts, bytes, and learned embeddings of categorical tags. For each edge $(u \rightarrow v) \in E_T$, define edge features $x_{u,v} \in \mathbb{R}^q$ such as RPC method class, payload size estimate, and network zone relation. A trace-level representation $z_T \in \mathbb{R}^m$ aggregates node and edge features under a permutation-invariant graph operator, possibly conditioned on topology and time.

Anomaly detection produces a score $A(T) \in \mathbb{R}_{\geq 0}$ where larger values indicate more anomalous traces relative to a baseline distribution. The baseline may be per-route, per-service, or per-cluster, and it may

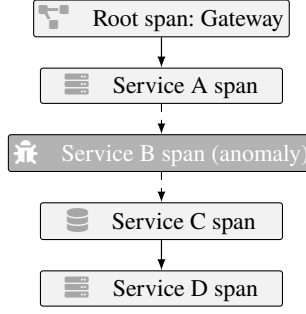


Figure 4: Trace fragment organized around the anomalous span, exposing its parents and children in the call chain as the structural context used for explanation.

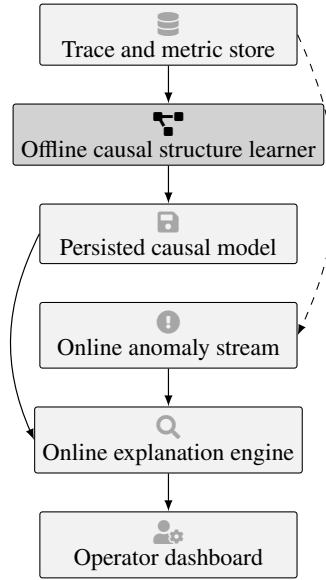


Figure 5: Integration of offline and online components: historical traces train a causal model that is reused in the live pipeline to provide operator-facing explanations for each detected anomaly.

change over time. For explanation, the paper assumes $A(T)$ is available and may be differentiable with respect to representation parameters, though it need not be differentiable with respect to raw system state. Explanations seek to identify a small set of variables S such that intervening on them would reduce $A(T)$ substantially. Because actual interventions are not performed, counterfactual reasoning is needed to estimate the effect [8].

To connect traces to causality, define a structural causal model over trace-derived variables. Let W denote exogenous variables capturing unobserved influences such as background load, kernel scheduling, or uninstrumented services. Let X denote endogenous variables derived from the trace, including span durations, queue times, fan-out counts, and routing decisions. The SCM specifies assignments $X_i := f_i(\text{Pa}(X_i), W_i)$ where $\text{Pa}(X_i)$ denotes parents of X_i in a causal graph G_C . The causal graph is constrained by G_T and time order: a span’s duration can be influenced by durations of downstream spans (via synchronous waits) and by resource contention influenced by concurrent work [9]. This yields a mixed causal structure where some effects propagate along call edges and some propagate through shared-resource confounders.

Dataset	#Services	#Traces (M)	Anomaly Rate (%)
TrainTicket	39	1.2	7.5
OnlineBoutique	11	0.8	5.2
DeathStarBench	24	1.6	9.1
SockShop	14	0.5	6.8
SyntheticTrace	20	2.0	10.0

Table 1: Benchmark datasets used for anomaly explanation experiments.

Method	Top-1 Accuracy	Top-3 Accuracy	MRR
Random Guess	0.11	0.29	0.21
KPI Correlation	0.34	0.55	0.43
OneHop Heuristic	0.41	0.63	0.49
GNN-RootCause	0.52	0.74	0.61
Temporal-Causal	0.58	0.79	0.66
CausalTrace (ours)	0.67	0.86	0.74

Table 2: Root cause ranking performance on distributed trace benchmarks.

Variant	Causal Graph	Counterfactual Ranking	Top-1 Accuracy
Full model	✓	✓	0.67
w/o causal graph	×	✓	0.59
w/o counterfactuals	✓	×	0.61
local-only features	✓	✓	0.62
no temporal edges	✓	✓	0.64

Table 3: Ablation study of key components in the proposed explainer.

#Services	#Edges	Inference Time (ms/trace)	Memory (MB)
10	40	4.2	120
20	95	7.8	180
40	210	13.6	260
80	460	25.1	410
120	720	38.7	560

Table 4: Scalability of the explanation pipeline with respect to service graph size.

A core modeling choice is to define causal variables at multiple granularities. Span-level variables capture local behavior but can be high dimensional and noisy. Service-level variables aggregate spans of the same service within a trace, reducing variance but risking ecological bias [10]. Route-level variables capture systemic behavior across traces. The framework supports all three by defining a hierarchy and allowing interventions at the level that matches operational actions, such as rolling back a deployment (service-level), changing a retry policy (route-level), or disabling a specific feature flag (tag-level).

Counterfactual explanation is posed as an optimization problem. Let $\mathcal{I}$ be a set of feasible interventions. Each intervention $I \in \mathcal{I}$ specifies a set of variables S_I and target values $\tilde{x}_{S_I}$, producing a modified SCM $M^{\text{do}(I)}$ under the do-operator. Let $T^{\text{cf}}(I)$ denote the counterfactual trace representation predicted under I , and define the counterfactual anomaly score $A^{\text{cf}}(I) = A(T^{\text{cf}}(I))$. The explanatory effect is

Metric	Rule-based	Black-box GNN	CausalTrace (ours)
Fidelity to model	3.1	3.7	4.4
Precision of causes	2.9	3.5	4.5
Conciseness	3.8	3.1	4.2
Actionability	3.0	3.4	4.6

Table 5: Expert evaluation scores (1–5) of explanation quality.

Service	#Parents	#Children	Avg. Inter-service Delay (ms)
Auth	0	2	4.3
User	1	3	5.7
Order	2	2	7.9
Payment	1	1	9.4
Inventory	1	2	6.1
Shipping	2	0	8.7

Table 6: Structural statistics of the learned causal graph on an e-commerce workload.

Rank Predicted Latency Change (ms)	Node / Span	Intervention
1 –82	Payment.DB write	limit retries to 1
2 –47	Inventory cache	increase TTL by 50%
3 –33	Auth token check	enable local validation
4 –21	Shipping service	batch size from 1 to 10
5 –15	Order aggregator	parallelize fan-in merge

Table 7: Top counterfactual interventions recommended for a misbehaving trace.

$\Delta(I) = A(T) - A^{\text{cf}}(I)$. Interventions also have a cost $C(I)$ representing operational effort, risk, or invasiveness [11]. The explanation task is to rank interventions by a utility that combines effect and cost while accounting for uncertainty.

A representative objective is

$$U(I) = \mathbb{E}[\Delta(I) | T] - \lambda C(I) - \gamma \text{Var}(\Delta(I) | T) \quad , \quad (2.1)$$

where λ and γ are trade-off parameters. The expectation and variance reflect epistemic uncertainty due to partial observability and approximation [12]. This formulation emphasizes that explanation is not simply a maximal effect problem; it is a constrained decision problem where robustness matters.

Because intervention spaces can be combinatorial, complexity and approximations are central. If interventions allow arbitrary subsets of variables, selecting a minimal subset that achieves a target reduction in anomaly score resembles sparse optimization and can encode NP-hard problems. Even when restricting to graph neighborhoods, the number of candidate explanations can be large in high-fan-out traces [13]. The framework therefore imposes structural constraints from G_C and uses

Component	Hyperparameter	Value	Description
Encoder	hidden dimension	128	span feature size
Encoder	dropout	0.2	regularization
Causal learner	sparsity λ	0.05	edge penalty
Causal learner	max lag	3	hops in time
Ranker	temperature τ	0.7	soft ranking
Optimizer	learning rate	1e-3	Adam step size

Table 8: Key hyperparameters of the causal explanation model.

Noise in Traces (%)	Top-1 Accuracy	MRR	Kendall- τ
0	0.67	0.74	0.81
10	0.65	0.72	0.79
20	0.62	0.69	0.76
30	0.58	0.65	0.72
40	0.53	0.60	0.68
50	0.47	0.54	0.61

Table 9: Robustness of root cause ranking under increasing trace corruption.

heuristics informed by gradients, graph centrality, and low-rank interaction structure to generate and rank candidates efficiently.

3. Causal Graph Construction from Distributed Traces

Constructing a causal graph from trace data must balance domain constraints, statistical evidence, and computational feasibility. The causal graph here is not assumed to represent the full physical reality of the distributed system; rather, it is an operational causal abstraction over trace-derived variables that supports counterfactual ranking.

A baseline constraint graph arises from call topology and time [14]. If span u is an ancestor of span v in G_T and the execution is synchronous, then the parent’s duration includes waiting on the child, suggesting a directed influence from v to u in terms of duration. Conversely, if u triggers v asynchronously, the influence may not be direct in duration but may exist in resource contention or downstream saturation. These differences can often be inferred from span kind, semantic tags, or observed overlap. The framework defines candidate causal edges based on three relations: synchronous wait edges, asynchronous trigger edges, and shared-resource edges induced by co-location or shared queues [15].

Let Y_s be the duration (or log-duration) of span s , and let Q_s be an estimated queueing component, such that $Y_s \approx Q_s + \sum_{c \in \text{Ch}(s)} Y_c + \epsilon_s$ in synchronous patterns, where $\text{Ch}(s)$ are children spans and ϵ_s captures local processing. A causal graph can be constructed with directed edges $Y_c \rightarrow Y_s$ for synchronous dependencies. Shared-resource confounding can be represented by latent variables L_r for resource r (CPU pool, thread pool, database connection pool, network link) with edges $L_r \rightarrow Y_s$ for spans affected by r . When explicit resource metrics are available, L_r can be partially observed as utilization or queue depth, turning latent confounders into measured variables and improving identifiability.

Statistical refinement is needed because topology alone can over-connect [16]. The framework uses conditional independence testing under constraints. For each candidate edge $i \rightarrow j$, it evaluates whether adding the edge improves predictive fit of X_j given a parent set $\text{Pa}(X_j)$, while penalizing complexity. Because trace data are heavy-tailed and nonstationary, the approach favors robust regression and quantile-aware loss functions. A simple but effective mechanism uses generalized linear models with Huber loss or quantile loss, combined with time-windowed normalization per route.

Graph learning is further stabilized by embedding-based similarity [17]. Each service-operation pair is assigned an embedding vector $e_k \in \mathbb{R}^d$ learned from co-occurrence in traces and from tag context. Given embeddings, candidate edges between variables associated with services a and b can be prioritized by cosine similarity $\text{sim}(a, b) = \frac{e_a^\top e_b}{\|e_a\| \|e_b\|}$, capturing that certain operations have consistent coupling across routes. Embeddings also support generalization under sparse data: new deployments with limited trace history can borrow structure from similar operations.

To control complexity, the interaction structure among services can be compressed via low-rank approximation. Define an empirical interaction matrix $M \in \mathbb{R}^{n \times n}$ where $M_{i,j}$ measures how often service i calls service j on a route and how much of the parent’s time is attributable to the child. This matrix is often approximately low-rank due to layered architectures and shared libraries. The framework computes a truncated SVD [18]

$$M \approx U_k \Sigma_k V_k^\top, \quad (3.1)$$

where $k \ll n$ is chosen under an energy criterion. This factorization provides compact latent factors that can be used both for candidate edge scoring and for approximating causal propagation computations. For example, influence scores that sum along call edges can be approximated by operations in the k -dimensional latent space, reducing per-trace overhead [19].

Causal semantics require careful treatment of selection and sampling. Trace sampling is rarely uniform; tail-based sampling oversamples slow traces, inducing selection bias. If anomaly detection and explanation are performed on sampled traces, naive estimators can exaggerate the importance of variables correlated with sampling probability. The framework incorporates inverse-propensity weighting when sampling policies are known or can be estimated [20]. Let $\pi(T)$ be the probability a trace is sampled. Weighted estimators replace averages $\mathbb{E}[\cdot]$ with $\mathbb{E}[\cdot/\pi(T)]$ in model fitting, and uncertainty is increased when $\pi(T)$ is small or poorly estimated. When $\pi(T)$ depends on observed duration, weighting can still be unstable; the framework therefore caps weights and propagates the induced bias into the variance term of the ranking objective.

A practical causal graph must support fast queries. The system stores G_C as adjacency lists keyed by variable identifiers, using compact integer encoding for service-operation pairs and variable types [21]. Because high-cardinality tags can explode node types, the framework restricts explicit nodes to tags that pass frequency and mutual information thresholds, and represents the rest via hashed feature projections. Hashed projections are used cautiously because they can create collisions that harm interpretability. The framework mitigates this by using multiple hash functions and tracking collision probability bounds. If h independent hashes map into B buckets, then for m active features the expected number of collisions scales roughly as $O(m^2/(hB))$, and the system chooses B to keep collision rates below a target such as 1% under typical m [?].

Graph algorithms are used to derive causal neighborhoods for candidate explanations. For a given anomalous trace, the system computes a subgraph induced by nodes whose observed values are extreme under baseline. Extremeness can be measured by robust z-scores, tail probabilities from fitted distributions, or reconstruction error under an autoencoder representation. The causal neighborhood is then expanded by following directed edges up to a bounded radius, using breadth-first traversal with pruning based on edge strength [22]. This yields a tractable candidate set even when the full causal graph is large.

4. Counterfactual Modeling and Ranking for Explanations

Counterfactual ranking requires a model that predicts how interventions on trace-derived variables would change the anomaly score. Because actual interventions are not performed during analysis, the model must extrapolate from historical variability and causal structure. The framework combines explicit causal mechanisms for some variables with learned conditional models for others, and uses uncertainty to avoid overconfident explanations [23].

The anomaly score $A(T)$ is treated as a function of a trace representation z_T . A common representation aggregates span features via message passing or pooling. To keep explanation interpretable, the framework constrains the representation to be decomposable by variables of interest. Let v_i denote a causal variable, such as duration of a specific service-operation within the trace, retry count on an edge, or queue depth metric [24]. The representation is constructed as

$$z_T = \sum_{i=1}^m \phi_i(v_i) \quad , \quad (4.1)$$

where ϕ_i are learnable maps into a shared latent space. Decomposability allows attribution and enables intervention modeling by modifying only the affected $\phi_i(v_i)$ terms. While full decomposability may reduce expressive power, it improves stability and interpretability [25]. When limited interactions are necessary, the framework adds low-rank pairwise terms constrained by the causal graph:

$$z_T = \sum_i \phi_i(v_i) + \sum_{(i,j) \in E_C} \psi_{i,j}(v_i, v_j) \quad , \quad \psi_{i,j}(v_i, v_j) = a_{i,j} \langle P_i v_i, P_j v_j \rangle \quad , \quad (4.2)$$

where P_i are projection matrices into a low-dimensional space and $a_{i,j}$ are edge weights. This yields an interaction model that remains computationally manageable and aligned with causal adjacency.

Counterfactual inference proceeds in three steps: abduction, action, and prediction. Abduction infers exogenous noise terms consistent with the observed trace under the SCM [26]. Action modifies the structural equations for intervened variables. Prediction computes the distribution of downstream variables under the modified model. In practice, full abduction over high-dimensional W is infeasible; the framework uses amortized inference where exogenous residuals are represented by per-variable residuals $\hat{\epsilon}_i$ inferred by regressing v_i on its parents and taking the residual. These residuals are treated as fixed in the counterfactual, a common approximation when noise is assumed to be independent and intervention does not change exogenous factors.

For a variable v_j with parents $\text{Pa}(v_j)$, the structural equation is modeled as

$$v_j = f_j(v_{\text{Pa}(j)}) + \epsilon_j \quad , \quad (4.3)$$

where f_j can be a robust linear model, a monotone spline, or a small neural network constrained for stability [27]. Under an intervention $\text{do}(v_s = \tilde{v}_s)$, the counterfactual value for a descendant v_j is obtained by topological propagation:

$$v_j^{\text{cf}} = f_j(v_{\text{Pa}(j)}^{\text{cf}}) + \hat{\epsilon}_j \quad , \quad (4.4)$$

with $v_s^{\text{cf}} = \tilde{v}_s$ and $v_k^{\text{cf}} = v_k$ for non-descendants outside the affected region, unless shared-resource nodes mediate broader effects. Because the causal graph may include latent confounders approximated by measured resource metrics, propagation can incorporate those metrics if they are included as parents. When confounders are unmeasured, the framework inflates uncertainty and penalizes explanations that rely on tenuous identification.

The key computational challenge is ranking many candidate interventions. Candidate generation is constrained to small sets, often single-variable or small multi-variable interventions [28]. Even with small sets, the number of candidates can be large when considering all anomalous variables. The framework uses a two-stage ranking: a fast approximate score for pruning and a slower counterfactual evaluation for the shortlist.

The fast score uses local sensitivity of the anomaly score to variables. If A is differentiable with respect to v_i through z_T , then a first-order approximation yields [29]

$$A^{\text{cf}}(I) \approx A(T) + \sum_{i \in S_I} \frac{\partial A}{\partial v_i} (\tilde{v}_i - v_i) \quad . \quad (4.5)$$

To maintain interpretability, gradients are computed with respect to normalized variables and are aggregated per service-operation. The framework also computes integrated gradients along a path from baseline values v_i^{base} to observed values v_i to reduce sensitivity to local nonlinearity:

$$\text{IG}_i = (v_i - v_i^{\text{base}}) \int_0^1 \frac{\partial A}{\partial v_i} (v_i^{\text{base}} + \alpha(v_i - v_i^{\text{base}})) d\alpha \quad . \quad (4.6)$$

These attribution-like quantities are used only for pruning; final explanations rely on counterfactual propagation in the SCM.

Candidate interventions specify target values $\tilde{v}_i$. Choosing $\tilde{v}_i$ as a baseline median can be too aggressive if the system operates in multiple modes. The framework instead uses conditional baselines: $\tilde{v}_i$ is chosen as a quantile of the conditional distribution of v_i given its parents, route, and context. Under a fitted conditional model, $\tilde{v}_i$ can be the conditional median or a value that corresponds to a target tail probability, such as moving from the 99% tail back to the 50% quantile. This choice supports explanations that correspond to realistic operational improvements rather than idealized states.

Ranking uses a multi-objective utility [30]. One operationally relevant formulation is a constrained optimization in which explanatory effect must exceed a threshold while cost is minimized:

$$\min_{I \in \mathcal{I}} C(I) \quad \text{subject to} \quad \mathbb{E}[\Delta(I) \mid T] \geq \tau \quad , \quad (4.7)$$

with τ chosen based on alert severity. Another formulation uses a weighted sum and includes computational overhead $L(I)$ as a latency term:

$$\max_{I \in \mathcal{I}} \mathbb{E}[\Delta(I) \mid T] - \lambda C(I) - \eta L(I) - \gamma \text{Var}(\Delta(I) \mid T) \quad . \quad (4.8)$$

Weights λ, η, γ can be tuned per team or derived from service-level objectives [31]. The Pareto frontier of candidates is also useful: candidates not dominated in both effect and cost can be presented to operators, providing options under uncertainty.

Computing $\mathbb{E}[\Delta(I) \mid T]$ requires integrating over model uncertainty. The framework supports Bayesian-ish modeling by placing approximate posterior distributions over parameters of f_j . For computational reasons, it uses variational approximations or ensembles rather than full MCMC. In an ensemble approach, K models are trained on bootstrapped windows and yield predictions $A_k^{\text{cf}}(I)$. Then

$$\mathbb{E}[\Delta(I) \mid T] \approx \frac{1}{K} \sum_{k=1}^K (A(T) - A_k^{\text{cf}}(I)) \quad , \quad \text{Var}(\Delta(I) \mid T) \approx \frac{1}{K-1} \sum_{k=1}^K (\Delta_k(I) - \bar{\Delta}(I))^2 \quad . \quad (4.9)$$

This captures both parameter uncertainty and some nonstationarity if bootstraps are time-aware [32].

The intervention selection problem can be NP-hard when multi-variable interventions are allowed with arbitrary costs. A reduction intuition arises from set cover: suppose each variable intervention fixes a subset of anomaly-contributing factors and costs are uniform; selecting a minimum number of variables to reduce anomaly below threshold resembles covering all factors. More formally, if the anomaly score decomposes as a sum of indicator-like penalties for violated constraints and each intervention can satisfy some constraints, then finding a minimal intervention set is at least as hard as minimum set cover, which

is NP-hard. The framework therefore uses heuristics [33]. A practical heuristic is greedy selection under diminishing returns. At step k , choose variable i that maximizes marginal gain per cost:

$$i^* = \arg \max_{i \notin S} \frac{\widehat{\Delta}(S \cup \{i\}) - \widehat{\Delta}(S)}{C(i)} , \quad (4.10)$$

where $\widehat{\Delta}$ is a fast counterfactual approximation computed on a local subgraph. Greedy is not optimal in general, but it is scalable and often adequate when causal influences are approximately submodular due to saturation effects in anomaly scoring. When submodularity is uncertain, the framework uses beam search on top of greedy, keeping a small beam width to manage complexity [34].

Dynamic programming appears in query planning for counterfactual propagation. If G_C restricted to a trace neighborhood is a tree or has bounded treewidth, then counterfactual values for descendants under an intervention can be computed efficiently by caching intermediate computations on subtrees. In practice, causal graphs in microservices can have loops at the service-interaction level due to retries and circuit breakers, but at the trace level the observed call graph is acyclic. Shared-resource nodes can reintroduce cross-links; the framework handles this by introducing separators and using iterative fixed-point updates with damping [35]. Complexity is controlled by limiting iterations and bounding neighborhood size.

Backprop-friendly derivatives are useful when learning parts of the SCM and anomaly model jointly. If A depends on parameters θ and f_j depend on parameters ϕ , then learning can optimize a joint objective that encourages counterfactual consistency:

$$\min_{\theta, \phi} \mathbb{E}_T \left[\ell_{\text{det}}(A_{\theta}(T), y_T) + \beta \ell_{\text{cf}}(A_{\theta}(T^{\text{cf}}(I)), A_{\theta}(T) - \Delta_{\text{target}}(I)) \right] , \quad (4.11)$$

where ℓ_{det} trains detection against labels y_T when available, and ℓ_{cf} regularizes counterfactual behavior under synthetic interventions I . Synthetic interventions can be drawn from historical regressions and recoveries, or from perturbations that mimic known failure modes [?]. This training improves ranking stability by aligning the anomaly score with causal propagation.

5. System Architecture, Storage Internals, and Distributed Execution

Operational explanation requires a pipeline that ingests traces, constructs features, maintains causal models, and serves ranked explanations with low latency. The architecture is designed around streaming ingestion with offline and online components [36].

Trace ingestion is handled by collectors that batch spans and export to a storage backend. A typical storage schema uses a write-optimized log for spans and secondary indexes for trace-id, service, and time. Explanation needs additional indexes to retrieve comparable baseline traces quickly and to access precomputed causal neighborhoods. The framework stores per-route baseline statistics in a columnar store keyed by route signature and time window [37]. Route signature is computed as a stable hash of normalized endpoint templates and selected tags, with care to avoid including high-cardinality identifiers. Hashing uses a consistent scheme so that route identity remains stable across rollouts. To limit collisions, a 64-bit hash is used with a secondary checksum.

Span features are computed in a streaming job [38]. Some features require joining spans within a trace, such as critical path estimates or fan-out degree. These are computed by buffering spans per trace-id until completion or timeout. Since buffering can be expensive under high throughput, the framework uses bounded-memory strategies: it limits per-trace buffered spans and evicts incomplete traces to a fallback path that computes partial features. Partial traces increase uncertainty; this is reflected in ranking by increasing $\text{Var}(\Delta(I) \mid T)$ and by penalizing explanations that rely on missing regions.

Graph representations for traces can be large, so the system stores compressed adjacency [39]. Parent-child relationships are stored as offsets into span arrays, enabling cache-friendly traversal. For each trace,

spans are sorted by start time; parent pointers allow reconstructing the tree. Additional inferred edges, such as shared-resource links, are not stored per trace but are implied by service co-location and resource identifiers. Causal graph G_C is stored separately as a service-operation-level graph with variable types, enabling reuse across traces and reducing per-trace footprint [40].

Approximate aggregation is critical at scale. Baseline estimation for variables uses sketches rather than storing all historical samples. For quantiles, the framework uses mergeable quantile sketches; for means and variances, it uses streaming estimators with decay. For heavy-tailed variables like duration, it maintains log-scale statistics and robust measures like median absolute deviation [41]. Sketch approximation introduces error. If $\hat{q}_\alpha$ is the estimated α -quantile from a sketch, then the quantile error influences target values $\tilde{v}_i$ and thus counterfactual scores. The system propagates this by adding a term to uncertainty:

$$\text{Var}(\Delta(I) \mid T) \leftarrow \text{Var}(\Delta(I) \mid T) + \sum_{i \in S_I} \left(\frac{\partial A^{\text{cf}}}{\partial \tilde{v}_i} \right)^2 \text{Var}(\hat{q}_{\alpha_i}) \quad , \quad (5.1)$$

where $\text{Var}(\hat{q}_{\alpha_i})$ is estimated from sketch properties and sample counts.

Online serving of explanations is implemented as a microservice that takes a trace-id and returns ranked candidates. The serving path executes a sequence of steps: load trace spans and features; compute anomaly decomposition and identify extreme variables; extract causal neighborhood from G_C ; generate candidates; compute approximate pruning scores; compute full counterfactual propagation for short-listed candidates; compute utility and rank; return an explanation object containing variables, causal paths, and predicted effects [42]. To meet latency targets, the service uses parallelism across candidates and caches baseline conditional models per route.

Computational complexity must be controlled. Let n be the number of spans in a trace and m the number of causal variables considered for explanation. Extracting a neighborhood in adjacency-list form is $O(m + |E_{\text{neigh}}|)$, with $|E_{\text{neigh}}|$ bounded by pruning. Candidate generation for single-variable interventions is $O(m)$ [43]. For multi-variable interventions with greedy selection of size k , the naive approach recomputes counterfactuals each step, leading to $O(km)$ evaluations. Each evaluation involves causal propagation over a neighborhood of size r , costing $O(r + |E_r|)$. The framework reduces this by caching descendant computations and using incremental updates when interventions are nested. If descendants are computed once and stored, then additional interventions can reuse unaffected subtrees, making marginal updates closer to $O(\Delta r)$ where Δr is the number of newly affected nodes [?].

Distributed execution arises in two places: training and serving. Training conditional models f_j and anomaly model parameters occurs in batch jobs over large trace datasets. Data partitioning is done by route signature and time window to preserve locality and reduce shuffle. Service-operation embeddings are trained using negative sampling over call edges; this can be distributed by partitioning edges and using asynchronous parameter updates [44]. Consistency is managed by periodic synchronization and by using bounded staleness, which is adequate because embeddings are used as priors rather than final explanations.

Serving can also be distributed by sharding traces across storage nodes and by using a stateless explanation service that fetches needed data. To reduce storage reads, the system caches recent anomalous traces and their computed candidate lists. Caching must account for nonstationarity: if baseline models update, cached explanations can become stale [45]. The framework tags cached results with model version identifiers and invalidates on version change.

Performance engineering includes memory layout, vectorization, and minimizing serialization overhead. Feature vectors are stored in contiguous arrays and operations like projection $P_i v_i$ use prepacked matrices to exploit SIMD. Graph traversal avoids recursion to reduce overhead and uses iterative stacks [46]. Counterfactual propagation uses topological ordering precomputed for the neighborhood. When fixed-point iterations are needed due to shared-resource cross-links, the system uses damping to ensure

convergence within a few iterations, with a fallback to returning a warning of elevated uncertainty if convergence is not achieved.

Operational constraints such as CPU budget and energy usage can be incorporated via Lagrangians. Suppose serving an explanation incurs cost $L(I)$ in milliseconds of CPU time and $E(I)$ in joules estimated from CPU cycles [47]. If the service has budgets $L_{\max}$ and $E_{\max}$, then the ranking and pruning can incorporate constraints:

$$\max_{I \in \mathcal{I}} \mathbb{E}[\Delta(I) | T] - \lambda C(I) \quad \text{subject to} \quad L(I) \leq L_{\max} \quad \text{and} \quad E(I) \leq E_{\max} \quad . \quad (5.2)$$

A relaxed form uses multipliers:

$$\max_{I \in \mathcal{I}} \mathbb{E}[\Delta(I) | T] - \lambda C(I) - \mu (L(I) - L_{\max})_+ - \nu (E(I) - E_{\max})_+ \quad , \quad (5.3)$$

where $(\cdot)_+$ penalizes budget violations. This encourages selecting explanations that are not only meaningful but also cheap to compute under load spikes.

6. Error Analysis, Approximate Queries, and Robustness

Distributed trace analytics often rely on approximate queries and lossy data: sampling, sketches, partial traces, and clock skew [48]. Explanation quality must therefore be analyzed under approximation error, because counterfactual ranking can amplify small estimation errors into incorrect prioritization.

Consider an estimated anomaly score $\hat{A}(T)$ computed from approximate aggregates, and a counterfactual estimate $\hat{A}^{\text{cf}}(I)$. The estimated effect is $\hat{\Delta}(I) = \hat{A}(T) - \hat{A}^{\text{cf}}(I)$. Let true scores under full data be $A(T)$ and $A^{\text{cf}}(I)$. Then the effect error is

$$\hat{\Delta}(I) - \Delta(I) = (\hat{A}(T) - A(T)) - (\hat{A}^{\text{cf}}(I) - A^{\text{cf}}(I)) \quad . \quad (6.1)$$

If approximation errors in observed and counterfactual evaluations are correlated, some error cancels; if they are uncorrelated, error accumulates. The framework exploits correlated evaluation by reusing the same sketch structures and approximations for both observed and counterfactual computations whenever possible, so that differences reflect interventions rather than approximation noise [49].

Sketch-based feature aggregation has known bounds. For example, if a count-min sketch estimates counts with additive error bounded by ϵN with probability $1 - \delta$, then features derived from counts inherit this error. When anomaly scoring uses normalized counts, the relative error can be larger for small denominators. The framework reduces this by smoothing denominators and by bounding contribution of small-count features through clipping [50]. For quantile sketches, error bounds depend on sketch size and merge patterns. The framework tracks effective sample size per route and inflates uncertainty when it is low.

Clock skew introduces error in duration decomposition into queue and service time. If parent and child spans are recorded on different hosts, start and end times can be skewed [51]. The framework avoids relying on absolute timestamps across hosts; it uses durations as primary and uses overlap only when time synchronization is verified. Queue time estimates are therefore treated as noisy derived variables. In the SCM, their structural equations include larger noise terms, reducing confidence in explanations that hinge solely on queue time shifts.

Sampling bias is addressed by weighting, but weighting can increase variance [52]. The framework uses stabilized weights:

$$w(T) = \frac{\min(\pi_{\max}, \pi(T))}{\pi(T)} \quad , \quad (6.2)$$

where $\pi_{\max}$ caps the inverse weight. This introduces bias but reduces variance, and the bias-variance trade-off is accounted for in uncertainty. When sampling depends strongly on the anomaly itself, identifiability can fail. In such cases, the framework marks explanations as low confidence and prefers candidates supported by invariant relationships across sampling regimes, such as deterministic call dependencies [53].

Robustness also concerns model misspecification. The SCM uses simplified functional forms that may not capture complex dynamics like cascading timeouts. The framework therefore incorporates sensitivity analysis: it perturbs model parameters within plausible ranges and evaluates the stability of candidate rankings. A candidate is considered robust if it remains near the top across perturbations [54]. Stability can be quantified by rank correlation across ensemble members or by measuring the probability that a candidate is in the top- k set.

Counterfactual validity requires respecting constraints. Some interventions are physically infeasible, such as setting a child span duration below a minimum dictated by network latency or cryptographic operations. The framework enforces feasibility through constraint sets [55]. If variable v_i has bounds $[l_i, u_i]$ and monotonic relations, then interventions must satisfy them. Constraints can be integrated via projection. Suppose an unconstrained target is $\tilde{v}_i^{(0)}$; then the feasible target is

$$\tilde{v}_i = \Pi_C(\tilde{v}_i^{(0)}) \quad , \quad (6.3)$$

where Π_C projects onto feasible set C . Projection can be implemented as clipping for simple bounds, or as solving a small constrained optimization when multiple variables must satisfy shared constraints, such as end-to-end latency budget. When multiple variables are adjusted to match a total latency constraint, the framework uses a Lagrangian: [56]

$$\min_{\tilde{v}_S} \sum_{i \in S} \alpha_i (\tilde{v}_i - v_i)^2 \quad \text{subject to} \quad \sum_{i \in \text{path}} \tilde{v}_i \leq B \quad , \quad (6.4)$$

leading to a closed-form adjustment when the constraint is active, with multipliers that distribute reductions across variables proportional to α_i^{-1} , reflecting which changes are more feasible.

An additional source of error is confounding by shared load. If many services slow simultaneously due to an upstream traffic surge, an explanation that points to one service may be misleading. The framework partially addresses this by including load indicators as variables, such as request rate and concurrency, and by modeling them as parents of multiple services. When load is unmeasured, the framework uses proxies such as ingress queue depth or front-end latency and increases uncertainty [57]. It also penalizes explanations that focus on nodes with high correlation to global slowdowns unless there is evidence of asymmetric effect along causal paths.

Similarity metrics help cluster anomalies and validate explanations. Each trace can be embedded into a vector space using z_T and additional summary features. Similarity between traces can be computed by cosine distance or by Mahalanobis distance under a baseline covariance [58]. PCA is used to identify dominant anomaly modes: given a matrix of centered trace representations $Z \in \mathbb{R}^{N \times m}$, PCA computes components that explain variance. If anomalies cluster along a component aligned with a known dependency, explanations can be validated by consistency across traces. Low-rank approximations also improve performance: representing Z by its rank- k approximation reduces noise and accelerates nearest-neighbor retrieval used to find baseline comparators.

Communication efficiency and storage cost can be analyzed through entropy-like considerations. Trace attributes are high-entropy due to unique IDs; explanation should avoid transmitting high-entropy fields [59]. The framework constructs explanation payloads as compact schemas containing service-operation identifiers, variable deltas, and causal path summaries. If H denotes entropy of raw trace attributes and H_{exp} the entropy of explanation payload, then compression ratio is roughly H_{exp}/H , and operationally the goal is to keep this low while preserving actionable information. The framework uses dictionary encoding for identifiers and transmits numerical values in quantized form where appropriate,

such as encoding durations in microseconds with variable-length integers. Quantization introduces error but is negligible relative to model uncertainty when quantization step is small compared to baseline variability.

7. Evaluation Methodology and Reproducibility Considerations

Evaluating interpretable anomaly explanation is challenging because ground truth causes are rarely labeled at fine granularity [60]. The framework therefore defines evaluation along multiple axes: fidelity to counterfactual outcomes, causal plausibility, interpretability, stability, and system overhead.

A controlled evaluation uses fault injection in staging or dedicated testbeds. Faults include latency injection on a service, error-rate increase, resource throttling, and dependency degradation. For each injected fault, traces are collected, anomalies are detected, and explanations are generated [61]. Ground truth is known at the intervention point, enabling measurement of how often the top-ranked explanation identifies the injected service or a causal ancestor/descendant within a small radius. Because real systems have propagation, it is acceptable for an explanation to identify a near-causal neighbor if it provides actionable localization. Metrics include top- k hit rate, mean reciprocal rank, and causal distance between explanation target and injected fault node in the dependency graph. Interpretability is measured by explanation size, such as number of variables and path length, with a preference for smaller explanations when fidelity is similar [62].

A production-like evaluation uses historical incidents, where partial ground truth exists in incident timelines and change logs. If a deployment rollback resolves an incident, then explanations that rank the deployed service highly are considered aligned. This is noisy because multiple actions can occur, but it provides realism. The framework aligns traces with change events using time windows and route signatures [63]. It evaluates whether explanations shift appropriately before and after mitigation, which tests counterfactual reasoning implicitly.

Stability evaluation measures whether explanations are consistent across similar traces within an incident. Given a set of anomalous traces $\{T_i\}$ in a time window, compute explanations and measure the variance of top-ranked services and variables. Excessive variance indicates sensitivity to noise. Stability can be quantified by entropy of the explanation distribution; lower entropy suggests consistent localization [64]. However, too-low entropy can indicate a bias toward certain popular services. The framework balances this by comparing stability within incident windows to stability during normal periods.

Overhead evaluation measures the additional cost of explanation relative to detection-only pipelines. Costs include CPU time, memory, storage reads, and network bytes [65]. The framework reports p50 and p99 latencies for explanation serving under load, as well as throughput. It also measures the effect of approximation choices on both cost and quality, generating Pareto curves. For example, increasing sketch size improves quantile accuracy but increases memory; limiting neighborhood radius reduces computation but may miss distal causes. These trade-offs are central to deployment decisions [66].

Reproducibility requires precise definition of data splits, baselines, and model versions. The framework specifies that baseline statistics and conditional models are trained on time windows preceding evaluation windows, avoiding leakage. It also requires storing sampling policies, instrumentation versions, and schema versions, because these affect features. Deterministic pipelines are preferred: random seeds are fixed, and nondeterministic distributed operations are minimized [67]. When ensembles are used, each member is defined by a deterministic bootstrap procedure.

Evaluation also considers explanation calibration. If the model outputs an estimated effect $\mathbb{E}[\Delta(I) | T]$ and uncertainty, then calibration checks whether interventions predicted to have large effects correspond to larger realized improvements when similar real mitigations occur. In staging, realized improvements can be measured by actually applying mitigations. In production, realized improvements can be approximated by comparing anomaly scores before and after an action, controlling for confounding factors like traffic changes [68]. Calibration is imperfect but can be assessed with reliability diagrams comparing predicted effect bins to observed reductions.

Finally, evaluation should consider failure cases. When causal identifiability is low due to missing spans or heavy confounding, the system should degrade gracefully by returning broader explanations with higher uncertainty rather than confidently mislocalizing. Measuring how often the system signals low confidence in ambiguous cases is part of safety in operations [69]. This is not safety in the physical sense but operational reliability: avoiding misleading guidance that could trigger risky changes.

8. Conclusion

This paper developed a framework for interpretable anomaly explanation in distributed traces by combining causal graphs with counterfactual ranking. The approach formalizes trace-derived variables within a structural causal model constrained by call topology, time, and resource-sharing patterns, enabling explanation candidates to be interpreted as interventions rather than correlations. Counterfactual inference is performed through approximate abduction and graph-based propagation, and candidates are ranked under a multi-objective utility that balances predicted anomaly reduction, operational cost, computational overhead, and uncertainty [70]. The framework integrates representation learning through decomposable embeddings and low-rank interaction terms to support scalability while retaining interpretable variable contributions. It addresses production constraints through storage-aware graph representations, sketch-based approximate aggregation, caching, and distributed execution strategies, and it provides error analysis that propagates approximation uncertainty into ranking. Evaluation methodology emphasizes controlled fault injection, incident-aligned retrospective analysis, stability, overhead, and reproducibility. The overall result is an operationally implementable path from anomalous trace detection to ranked, causally motivated explanations, with explicit handling of uncertainty and trade-offs that arise in large-scale, evolving microservice environments [71].

References

- [1] S. Durga, P. Gupta, L. Kharb, P. Ranjit, V. H. R. Dornadula, K. C. Modak, and G. Manoharan, “9leveraging distributed systems for improved educational planning and resource allocation,” 3 2024.
- [2] M. Dahlmanns, F. Heidenreich, J. Lohmöller, J. Pennekamp, K. Wehrle, and M. Henze, “Unconsidered installations: Discovering iot deployments in the ipv6 internet,” in *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, pp. 1–8, IEEE, 5 2024.
- [3] O. J. Dellar and B. Jones, “Efficient frequency response computation for low-order modelling of spatially distributed systems,” *International Journal of Control*, vol. 93, pp. 366–376, 5 2018.
- [4] J. Tao, L. Xu, and S. Guo, “A deinterleaving approach of pulse signals in distributed system,” *Signal, Image and Video Processing*, vol. 19, 9 2025.
- [5] . Ευθυμιάδης and . Χαλδαίος, “Ανάπτυξη κατανεμημένου συστήματος σε υπολογιστικό νέφος,” 6 2018.
- [6] M. Zaccarini, F. Poltronieri, C. Stefanelli, and M. Tortonesi, “Hybridized hot restart via reinforcement learning for microservice orchestration,” in *NOMS 2025-2025 IEEE Network Operations and Management Symposium*, pp. 1–7, IEEE, 5 2025.
- [7] R. Malik, S. Kim, X. Jin, C. Ramachandran, J. Han, I. Gupta, and K. Nahrstedt, “Mlr-index: An index structure for fast and scalable similarity search in high dimensions,” in *International Conference on Scientific and Statistical Database Management*, pp. 167–184, Springer, 2009.
- [8] K. Johari, G. K. Agarwal, and Y. D. Arya, “Server load balancing analysis for client task assignment in distributed systems,” *ECS Transactions*, vol. 107, pp. 17605–17621, 4 2022.
- [9] *OSDI - Scalable Memory Protection in the PENGLOAI Enclave*, 6 2021.
- [10] D. Xenakis, E. Samikwa, J. Ajayi, A. D. Maio, T. Braun, and K. Schlegel, “Towards personality detection and prediction using smartphone sensor data,” in *2023 21st Mediterranean Communication and Computer Networking Conference (MedComNet)*, pp. 121–130, IEEE, 6 2023.

- [11] O. Savenko, A. Sachenko, S. Lysenko, G. Markowsky, and N. Vasylykiv, "Botnet detection approach based on the distributed systems," *International Journal of Computing*, pp. 190–198, 6 2020.
- [12] P. Schmidt, A. Jaust, H. Steeb, and M. Schulte, "Simulation of flow in deformable fractures using a quasi-newton based partitioned coupling approach," *Zenodo (CERN European Organization for Nuclear Research)*, 4 2021.
- [13] R. Alfonso, C. Daher, M. Arzamendia, K. Cikel, D. Gregor, D. Gutierrez, S. Toral, and M. Villagra, "A motorcyclist helmet detection system through a two-stage cnn approach," in *2021 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pp. 1–6, IEEE, 12 2021.
- [14] *On Strict Homogeneous Lyapunov Function for Generalized Homogeneous PI Controller*, 9 2021.
- [15] M. Witter and A. R. D. Vit, "Blockchain e sistemas distribuídos: conceitos básicos e implicações," 3 2024.
- [16] B. Sedlak, V. C. Pujol, P. K. Donta, and S. Dustdar, *Designing Reconfigurable Intelligent Systems with Markov Blankets*, pp. 42–50. Germany: Springer Nature Switzerland, 11 2023.
- [17] K. Habibi, "Web service replica selection analysis using a multiagent-based simulator," 5 2021.
- [18] L. Kroll, "Compile-time safety and runtime performance in programming frameworks for distributed systems," 3 2020.
- [19] E. Bazgir, T. T. Tanha, A. A. Bhuiyan, E. Haque, and M. S. Uddin, "Analysis of distributed systems," *International Journal of Computer Applications*, vol. 186, pp. 16–21, 11 2024.
- [20] M. Goudarzi, M. Palaniswami, and R. Buyya, "A distributed deep reinforcement learning technique for application placement in edge and fog computing environments," *IEEE Transactions on Mobile Computing*, vol. 22, pp. 2491–2505, 5 2023.
- [21] C. R. F. Campusano, "Distributed eventual leader election in the crash-recovery and general omission failure models.," 1 2020.
- [22] V. Vijaykumar, R. Chandrasekar, and T. Srinivasan, "An obstacle avoidance strategy to ant colony optimization algorithm for classification in event logs," in *2006 IEEE Conference on Cybernetics and Intelligent Systems*, pp. 1–6, IEEE, 2006.
- [23] R. Achar, P. Dawn, and C. V. Lopes, "Onward! - gotcha: an interactive debugger for got-based distributed systems," in *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, pp. 94–110, ACM, 10 2019.
- [24] H. H. Ali and A. D. S. H. Shaker, "Techniques for secure distributed systems," *Journal of Physics: Conference Series*, vol. 1530, pp. 012006–, 5 2020.
- [25] C. Marcelino, S. Gollhofer-Berger, T. Pusztai, and S. Nastic, "Cosmos: A cost model for serverless workflows in the 3d compute continuum," in *2025 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 106–113, IEEE, 6 2025.
- [26] M. Zaki, "Balancing privacy, precision and performance in distributed systems," 1 2020.
- [27] A. Rogan, A. Kolar-Požun, and G. Kosec, "A numerical study of combining rbf interpolation and finite differences to approximate differential operators," in *2025 MIPRO 48th ICT and Electronics Convention*, pp. 1177–1182, IEEE, 6 2025.
- [28] N. Phu, "Clara algorithm in a distributed system," 9 2018.
- [29] G. B. Nardes, W. Grignani, T. H. Rausch, L. F. H. Duarte, D. R. Melo, and C. A. Zeferino, "Evaluating fpga architectures for quantized cnn inference in construction monitoring applications," in *2025 32nd IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 1–4, IEEE, 11 2025.
- [30] M. Abebe, B. Glasbergen, and K. Daudjee, "Sigcse - watdfs: A project for understanding distributed systems in the undergraduate curriculum," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pp. 920–926, ACM, 2 2019.
- [31] J. Slak and G. Kosec, "Fast generation of variable density node distributions for mesh-free methods," *WIT transactions on engineering sciences*, vol. 122, pp. 163–173, 9 2018.
- [32] Y. Wang, X. Zhu, R. Mahajan, and S. Wang, "Programmable and adaptive scheduling for distributed systems," in *Proceedings of the 24th ACM Workshop on Hot Topics in Networks*, pp. 193–200, ACM, 11 2025.

- [33] Y. Dong and Z. Mi, “Ioguard: Software-based i/o page fault handling with one cpu core,” in *Proceedings of the 15th Asia-Pacific Symposium on Internetware*, pp. 337–346, ACM, 7 2024.
- [34] “Ieee distributed systems online,” 3 2023.
- [35] M. Dahlmanns, C. Sander, R. Decker, and K. Wehrle, “Secrets revealed in container images: An internet-wide study on occurrence and impact,” in *Proceedings of the ACM Asia Conference on Computer and Communications Security*, pp. 797–811, ACM, 7 2023.
- [36] R. Gorantla, S. S. P. D. Pantham, and H. Venkatesh, *Treasure Hunt on Web-Predicated Distributed Systems*, pp. 89–99. Book Publisher International (a part of SCIENCEDOMAIN International), 5 2022.
- [37] R. Chandrasekar and T. Srinivasan, “An improved probabilistic ant based clustering for distributed databases,” in *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI*, pp. 2701–2706, 2007.
- [38] I. Beschastnikh, P. Liu, A. Xing, P. Wang, Y. Brun, and M. D. Ernst, “Visualizing distributed system executions,” *ACM Transactions on Software Engineering and Methodology*, vol. 29, pp. 1–38, 3 2020.
- [39] V. C. Pujol, P. K. Donta, A. Morichetta, I. Murturi, and S. Dustdar, *Distributed Computing Continuum Systems – Opportunities and Research Challenges*, pp. 405–407. Germany: Springer Nature Switzerland, 3 2023.
- [40] K. W. HamaAli and S. R. M. Zeebaree, “Resources allocation for distributed systems: A review,” 4 2021.
- [41] F. F. da Silva and S. Wehner, “Entanglement improves coordination in distributed systems,” in *Proceedings of the 2nd Workshop on Quantum Networks and Distributed Quantum Computing*, pp. 14–20, ACM, 9 2025.
- [42] Y. Qian, S. Ying, and B. Wang, “Smc - anomaly detection in distributed systems via variational autoencoders,” in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2822–2829, IEEE, 10 2020.
- [43] A. U. Islam, M. Yang, A. Iqbal, A. Xu, S. Chen, and S. Asghar, “Trustworthy distributed systems: A machine learning approach,” in *2025 IEEE 20th Conference on Industrial Electronics and Applications (ICIEA)*, pp. 1–6, IEEE, 8 2025.
- [44] C. Zhu, W. Zheng, X. Fan, X. Deng, S. Liu, L. Yi, W. Xi, and Y.-S. Jeong, “Graph-empowered multidimensional target full-coverage reliability for internet of everything,” *IEEE Internet of Things Journal*, vol. 12, pp. 3707–3719, 2 2025.
- [45] M. Adham, S. J. Keene, T. Slay, J. T. Kolln, and R. B. Bass, “Grid services demonstration using service-oriented derms,” in *2025 IEEE PES Grid Edge Technologies Conference & Exposition (Grid Edge)*, pp. 1–5, IEEE, 1 2025.
- [46] B. Maier, D. Schneider, M. Schulte, and B. Uekermann, *Bridging scales with volume coupling — Scalable simulations of muscle contraction and electromyography*, pp. 185–199. Springer International Publishing, 1 2023.
- [47] G. Heinrich and I. Frosio, “Metaoptimization on a distributed system for deep reinforcement learning,” 2 2019.
- [48] H. Bouyghf, S. Abi, B. Benhala, and A. Raihani, “A comparison of metaheuristic techniques for rf passive components optimization,” 11 2018.
- [49] D. Ullmann, S. Rezaeifar, O. Taran, T. Holotyak, B. Panos, and S. Voloshynovskiy, “Information bottleneck classification in extremely distributed systems,” *Entropy (Basel, Switzerland)*, vol. 22, pp. 1237–, 10 2020.
- [50] M. Nosrati and M. Fazlali, “Community-based replica management in distributed systems,” *International Journal of Web Information Systems*, vol. 14, pp. 41–61, 4 2018.
- [51] Y. Zhao, W. Xue, W. Chen, W. Qiang, D. Zou, and H. Jin, “Owl: Differential-based side-channel leakage detection for cuda applications,” in *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, vol. abs/2209.14952, pp. 362–376, IEEE, 6 2024.
- [52] R. Chandrasekar, R. Suresh, and S. Ponnambalam, “Evaluating an obstacle avoidance strategy to ant colony optimization algorithm for classification in event logs,” in *2006 International Conference on Advanced Computing and Communications*, pp. 628–629, IEEE, 2006.
- [53] C. Stach, “Special issue on security and privacy in blockchains and the iot,” *Future Internet*, vol. 14, pp. 317–317, 11 2022.
- [54] T. Pollinger, A. V. Craen, C. Niethammer, M. Breyer, and D. Pflüger, “Leveraging the compute power of two hpc systems for higher-dimensional grid-based simulations with the widely-distributed sparse grid combination technique,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–14, ACM, 11 2023.

- [55] H. Mansouri and A.-S. K. Pathan, *SSCC - A Resilient Hierarchical Checkpointing Algorithm for Distributed Systems Running on Cluster Federation.*, pp. 99–110. Germany: Springer Singapore, 4 2020.
- [56] K. M. Goeschka, R. Oliveira, P. Pietzuch, and G. Russello, “Session details: Theme: Distributed systems: Dads - dependable, adaptive, and secure distributed systems track,” in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, ACM, 3 2020.
- [57] “Trusted working copies for distributed systems engineering,” 1 2022.
- [58] J. D. L. García and S. Imle, “A novel diagnosis system concept for safe distributed systems,” *Tecnología en Marcha*, vol. 34, pp. 19–27, 11 2021.
- [59] R. Yadav and R. Gaurkar, “Migration from virtualization to dockerization in cloud,” *International Journal of Science and Research (IJSR)*, vol. 13, pp. 545–548, 6 2024.
- [60] T. Srinivasan, R. Chandrasekar, V. Vijaykumar, V. Mahadevan, A. Meyyappan, and A. Manikandan, “Localized tree change multicast protocol for mobile ad hoc networks,” in *2006 International Conference on Wireless and Mobile Communications (ICWMC'06)*, pp. 44–44, IEEE, 2006.
- [61] A. Fentis, C. Lytridis, V. G. Kaburlasos, E. Vrochidou, T. Pachidis, E. Bahatti, and M. Mestari, “A machine learning based approach for next-day photovoltaic power forecasting,” in *2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS)*, pp. 1–8, IEEE, 10 2020.
- [62] G. A. Qadir and S. R. M. Zeebaree, “Evaluation of qos in distributed systems: A review,” 5 2021.
- [63] A. Samanta, C. Chetri, D. Karnehm, A. Neve, and S. Williamson, “Temporal sensitivity analysis of internal temperature informed charging algorithms and rapid thermal management system for e-mobility,” in *2024 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 2302–2304, IEEE, 10 2024.
- [64] M. Witter and A. R. D. Vit, “Blockchain and distributed systems: Basic concepts and implications,” 3 2024.
- [65] N. Xia, “Design strategy and case study of distributed system resilience in the chinese context,” *Strategic Design Research Journal*, vol. 13, pp. 355–363, 12 2020.
- [66] J. Szygula, P. Biernacki, D. Marek, A. Domanski, L. Sobczak, J. Flak, D. Caban, and P. Pawlas, “Analysis of web-based geo-visualization methods applied for automated guided vehicle using satellite navigation systems,” in *2022 IEEE International Conference on Big Data (Big Data)*, pp. 6371–6377, IEEE, 12 2022.
- [67] A. R. Pratama, F. J. Simanjuntak, A. Lazovik, and M. Aiello, *APPIS - Low-power Appliance Recognition using Recurrent Neural Networks*. 3 2018.
- [68] R. Malik, C. Ramachandran, I. Gupta, and K. Nahrstedt, “Samera: a scalable and memory-efficient feature extraction algorithm for short 3d video segments,” in *IMMERSCOM*, p. 18, 2009.
- [69] S. Agarwal, M. A. Rodriguez, and R. Buyya, “Input-based ensemble-learning method for dynamic memory configuration of serverless computing functions,” in *2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC)*, pp. 346–355, IEEE, 12 2024.
- [70] S. Sakr and A. Y. Zomaya, *Distributed Systems*, pp. 690–690. Springer International Publishing, 2 2019.
- [71] E. D. Caro, M. Brina, N. Belletti, F. Poltronieri, M. Tortonesi, and C. Stefanelli, “Timegraph: Synthetic generation of graph sequences for realistic mobile connectivity models,” in *2025 IEEE 11th International Conference on Network Softwarization (NetSoft)*, pp. 249–256, IEEE, 6 2025.