ORIGINAL ARTICLE

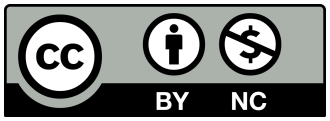
A Distributed Outlier Detection Method for High-Dimensional Telemetry Using Sketches and Locality-Sensitive Hashing

Naveen Perera[1] and Isuru Jayasena[2]

[1]Independent
[2]University of Ruhuna, Department of Electronics and Communication Engineering, Wellamadama Campus, Matara, Sri Lanka

ABSTRACT

Modern observability stacks collect high-dimensional telemetry that mixes numeric metrics, sparse categorical signals, and derived features from logs and traces. Detecting outliers in such data is operationally important but computationally constrained, since telemetry arrives as a distributed stream and must be processed under tight latency and bandwidth budgets. This paper develops a distributed outlier detection method that combines linear sketches for compact representation with locality-sensitive hashing to generate cross-node candidate neighborhoods. Each worker maintains a sliding-window index of sketched telemetry points, periodically exchanging small hash and sketch summaries rather than raw vectors. A coordinator (or peer-to-peer overlay) merges hash collisions into candidate sets and performs approximate neighborhood scoring using only sketches, while adaptively requesting extra sketch detail when uncertainty remains. The method targets distance- and density-based outlier notions in high dimensions, where exact nearest-neighbor search is impractical and naive aggregation is bandwidth-dominated. We formalize the distributed streaming setting, specify sketch constructions that preserve norms and approximate distances with controllable distortion, and couple them to Euclidean and angular LSH families for candidate generation. Analytical results characterize detection error as a function of sketch dimension, hash parameters, window size, and nonstationarity, yielding explicit trade-offs between false alerts and communication. Empirical evaluation on production-like telemetry mixtures and controlled synthetic drift scenarios indicates that sketch-only scoring recovers most of the ranking performance of raw-vector baselines while reducing per-window communication by one to two orders of magnitude under typical parameterizations.



1 | Introduction

Telemetry pipelines increasingly represent systems as large collections of interacting components that emit streams of measurements and events [1]. A single service instance may generate a vector that includes resource counters, timing statistics, quantiles, error codes, histogram bins, sparse one-hot encodings of categorical tags, and embeddings derived from log templates or trace spans. In practice, these vectors are high-dimensional, partially sparse, and nonstationary, with distributions shifting under deployments, autoscaling, and workload changes. Outlier detection in this environment is often framed as identifying rare patterns that correspond to faults, regressions, misconfigurations, or external incidents [2]. The operational constraints differ from classical offline anomaly detection: the data arrive as a distributed stream across many nodes, memory per node is limited, and the marginal cost of communication can exceed the marginal cost of local computation. Additionally, the definition of an outlier is typically relative to recent behavior rather than a fixed training distribution, motivating sliding-window or exponentially weighted formulations.

High dimensionality complicates both modeling and computation [3]. Distance concentration can make naive distance-based methods unstable, yet many robust detectors still rely on neighborhood structure, either explicitly through approximate nearest neighbors or implicitly through density estimation. In distributed telemetry, neighborhood structure is global: a point that appears normal on one node may be anomalous relative to the fleet, and conversely a rare event may be common within a localized region. Centralizing all raw telemetry to build global neighborhoods is often infeasible due to bandwidth, storage, privacy, and latency considerations [4]. Even when centralization is possible, the computational cost of exact neighbor search across millions of points in high dimension is prohibitive without specialized indexing.

This paper proposes a distributed method that couples two complementary ideas. Linear sketches provide compact, streaming-friendly representations that approximately preserve norms and distances while being mergeable and amenable to incremental updates. Locality-sensitive hashing (LSH) provides a principled mechanism for generating small candidate sets that are likely to contain near neighbors under a chosen metric [5]. The combination enables a workflow in which each worker compresses each telemetry vector into a sketch, inserts it into local LSH tables, and periodically exchanges only hash identifiers and small sketch summaries. Candidate neighborhoods across workers are produced by hash collisions, after which approximate distance computations can be performed directly on sketches. When needed, the protocol can request additional sketch components or secondary projections for ambiguous cases, yielding an adaptive accuracy-communication trade-off [6].

The contributions are methodological and analytical. Methodologically, we specify a sketched representation tailored to telemetry mixtures, with separate handling of sparse categorical features, dense numeric features, and heavy-tailed components. We define a distributed protocol that supports sliding windows, bounded memory, and partial synchronization, and we provide practical mechanisms for de-duplication, late arrivals, and drift-aware scoring [7]. Analytically, we relate the distortion of sketch-based distances to the collision probabilities of LSH and to the induced error in k-nearest-neighbor and local density scores. The resulting bounds suggest how to choose sketch dimension and LSH parameters as a function of the desired alert budget and bandwidth constraints, while highlighting failure modes under extreme nonstationarity or adversarial feature scaling.

2 | Background and Preliminaries

We consider telemetry points as vectors in a feature space that combines continuous and discrete components [8]. A common representation maps categorical keys and values to a high-dimensional sparse vector via hashing tricks, while continuous features occupy a lower-dimensional dense block. The combined representation yields a vector $x \in \mathbb{R}^d$ where d can be large, with most mass concentrated in a small subset of coordinates for any given point. In this setting, similarity metrics are chosen for robustness and interpretability. Euclidean distance is natural for standardized continuous metrics and for hashed one-hot vectors under appropriate scaling, while cosine

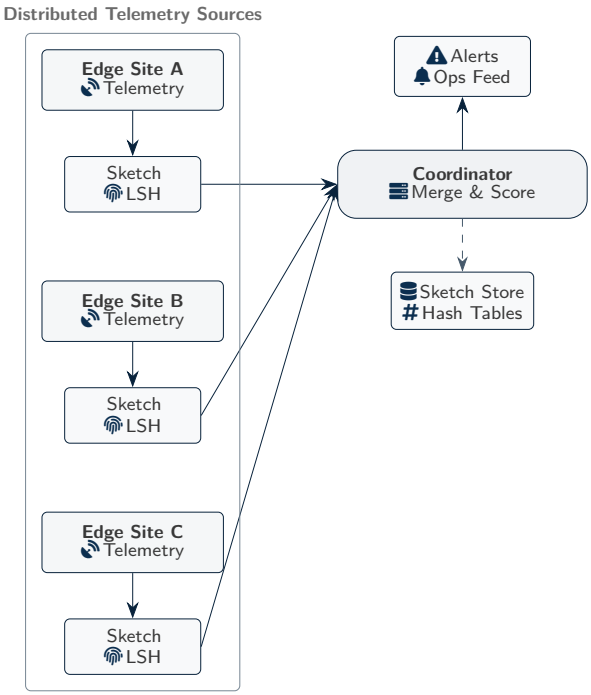


Figure 1: Distributed architecture for high-dimensional telemetry outlier detection: each edge site compresses features into sketches and LSH summaries, then transmits compact updates to a coordinator that merges evidence and emits alerts.

Symbol	Description	Type	Range
d	Telemetry dimensionality	Integer	10^2–10^4
n	Total number of data points	Integer	10^6–10^9
m	Sketch size per node	Integer	10^2–10^4
K	Number of LSH hash functions	Integer	1–64
L	Number of LSH tables	Integer	1–128

Table 1: Key notation used in the distributed outlier detection method.

similarity is often preferred when magnitude varies significantly across time or across services.

Outliers can be defined in multiple ways [9]. This paper focuses on neighborhood-based notions because they adapt to multimodality and do not require explicit parametric assumptions. A point is an outlier if its distance to its kth nearest neighbor is large relative to typical points in the same window, or equivalently if its estimated local density is low. For a window $\mathcal{W}$ of recent points, one can define the raw kNN radius $\rho_k(x)$ as the distance from x to the kth closest point in $\mathcal{W} \setminus \{x\}$. A score based on average neighbor distance is also common and can be more stable than a single order statistic [10]. In distributed settings, $\mathcal{W}$ is partitioned across workers, so both the neighbor search and the score normalization must incorporate cross-worker information.

Linear sketches are randomized linear maps or streaming summaries that reduce dimension while approximately preserving quantities of interest. For neighborhood scoring, the essential requirement is a map $S : \mathbb{R}^d \to \mathbb{R}^m$ such that pairwise distances are preserved up to small multiplicative distortion with high probability, while allowing streaming computation and small memory. Random projections inspired by Johnson–Lindenstrauss type embeddings meet this goal, and sparse variants reduce computation. In parallel, sketches such as Count-Min or CountSketch summarize sparse vectors and approximate coordinate magnitudes or norms, which can be useful for feature normalization, heavy-hitter identification, and robust scaling [11].

Locality-sensitive hashing constructs hash functions for which collision probability is higher for nearby points

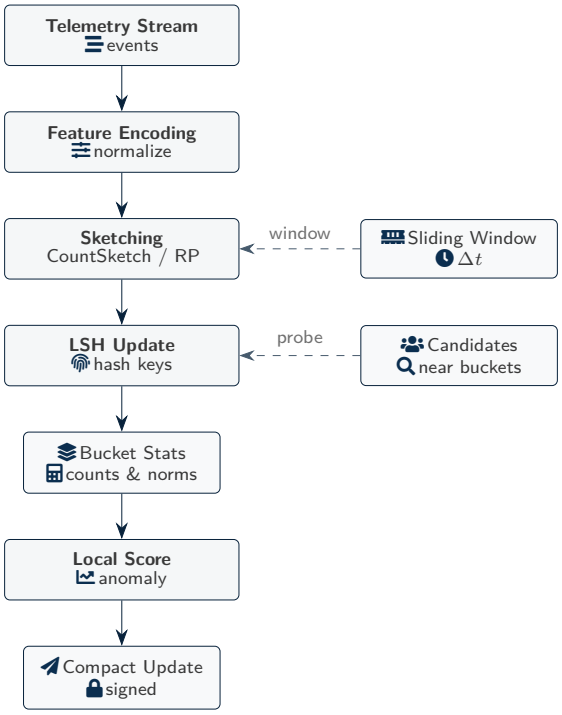


Figure 2: Per-node pipeline: telemetry is encoded, compressed via sketches, indexed with LSH, and summarized into bucket statistics that yield a local anomaly score and a compact signed update for distributed fusion.

Dataset	Points ($\times 10^6$)	Dim. d	Anomaly frac. (%)	Nodes
Synthetic-HiD	10.0	2,048	1.0	16
Telco-Net	4.2	512	0.3	32
Cloud-Metrics	7.5	1,024	0.5	64
IoT-Grid	3.0	256	2.0	20
Backbone-Flow	5.8	1,536	0.8	48

Table 2: Telemetry datasets used to evaluate the proposed method.

than for far points, under a specified metric. For cosine similarity, random hyperplane hashing maps vectors to bit signatures whose Hamming distance relates to the angle between vectors. For Euclidean distance, p-stable LSH uses random projections with stable distributions and quantization to buckets [12]. LSH is attractive in high dimension because it trades exactness for sublinear candidate generation, and because hash identifiers are cheap to communicate relative to raw vectors. In distributed telemetry, however, naive LSH still risks high overhead because one must either ship full hash tables to a coordinator or query all workers for each new point. The method developed here addresses this by aggregating bucket-level summaries and using sketches for scoring, so that full vector exchange is avoided.

3 Problem Formulation and Distributed Streaming Model

We model a fleet as N workers indexed by $i \in \{1, \dots, N\}$. Worker i observes a stream of telemetry points $\{x_{i,t}\}_{t \geq 1}$, where each point has a timestamp and is mapped into $\mathbb{R}^d$ after feature extraction and normalization. The global stream is the union of all worker streams [13]. We assume a sliding time window of width T (or an equivalent count-based window of size W) that defines the reference set for neighborhood scoring. For a query point x arriving at time t, the relevant comparison set is

$$\mathcal{W}(t) = \{x_{j,u} : t - T \leq u \leq t,\ 1 \leq j \leq N\}. \tag{1}$$

The goal is to output an outlier score $s(x)$ and optionally a binary decision $s(x) \geq \tau$, where τ is set to

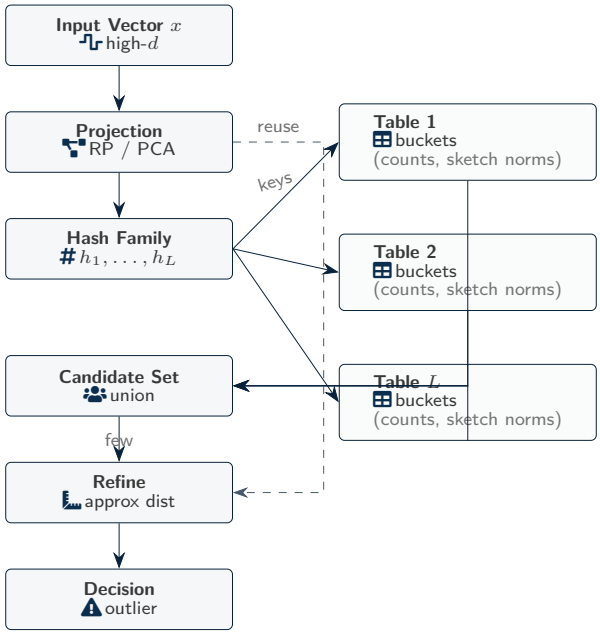


Figure 3: LSH indexing and query: projected vectors generate L hash keys to retrieve nearby-bucket candidates; refinement uses sketch-based approximate distances to decide whether the point is an outlier.

Node type	Sketch structure	Size per sketch (KB)	Sketches / node	Update time (μs)
Edge sensor	Count-sketch	64	4	3.1
Access router	Count-min	128	8	4.8
Aggregation switch	Hybrid (CM + CMS)	256	4	6.5
Core router	Tensor sketch	512	2	8.9
Coordinator	Aggregated view	1,024	1	15.2

Table 3: Sketch configurations across different node roles in the telemetry hierarchy.

satisfy an alert rate constraint [14]. The key constraint is that the system must operate online with bounded local memory and limited network communication. The distributed constraints are captured by a message budget. Each worker can periodically send messages of bounded size to a coordinator or to a subset of peers [15]. Messages may be delayed, and workers may drop messages under congestion. The protocol should tolerate such partial synchronization and should degrade gracefully when cross-worker information is incomplete. Computation at each worker should be close to linear in the number of observed points per window times the sketch dimension, and the coordinator's computation should be near linear in the number of reported candidate collisions rather than in the total number of points [16].

We assume that the feature mapping produces vectors that are approximately comparable across workers after standardization. In telemetry this is nontrivial because different services emit different metric families, and different deployments may change feature distributions. We incorporate two normalization stages. First, each worker maintains robust local scalers for its continuous features, such as running medians and median absolute deviations computed over the window, to reduce sensitivity to heavy tails [17]. Second, a fleet-level normalization is approximated by exchanging small sketches of feature moments, enabling approximate alignment of scales without transmitting raw values. The detection method remains neighborhood-based, but these normalization stages reduce distortions that would otherwise dominate the distance computation.

An adversarial threat model is not the focus, but the protocol should be resilient to benign pathologies such as bursty traffic, correlated failures, and concept drift [18]. Drift enters because what is normal at time t may be abnormal at time $t + \Delta$, and the window must

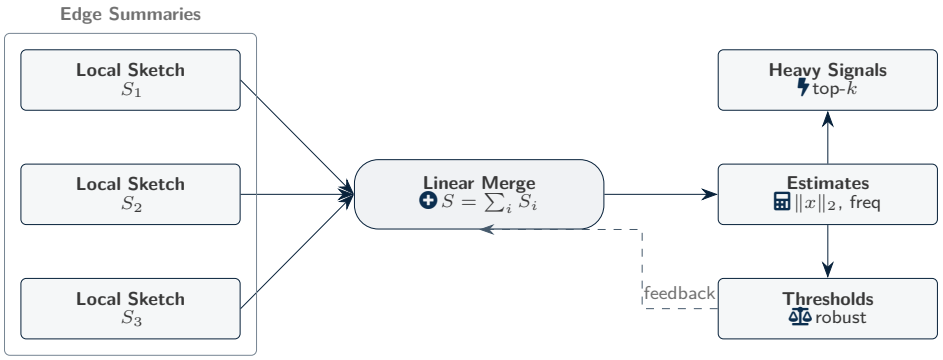


Figure 4: Sketch aggregation: linearity enables merging local sketches into a global sketch that supports approximate norm/frequency estimates, heavy-signal identification, and robust threshold calibration.

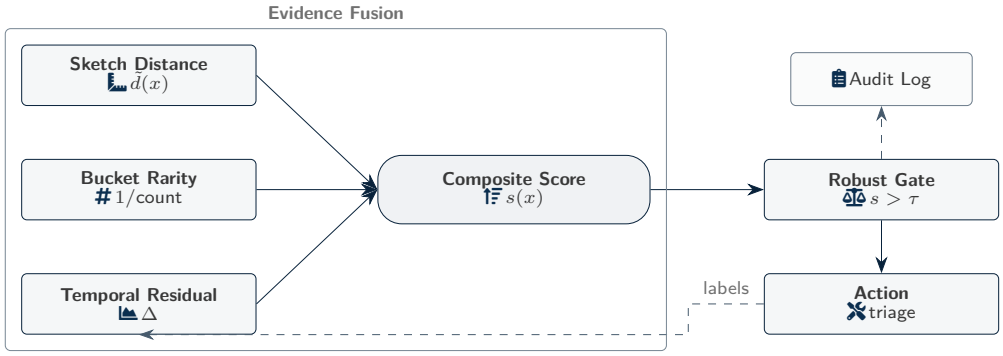


Figure 5: Outlier scoring: approximate sketch distance, LSH bucket rarity, and temporal residuals are fused into a composite score that is robustly gated into alerts, audit records, and optional feedback labels.

balance responsiveness with statistical stability. The protocol should also handle duplicates and near-duplicates that occur when the same event is observed by multiple workers or when instrumentation emits redundant metrics, as such duplicates can artificially inflate local density and mask anomalies. Deduplication is addressed by hashing stable identifiers and by using sketch-based similarity thresholds to collapse near-duplicates before neighborhood scoring [19].

4 | Sketch-Based Representation for High-Dimensional Telemetry

The sketching layer transforms each telemetry vector into a compact representation that supports approximate distance computations, robust scaling, and mergeable statistics. Let $x \in \mathbb{R}^d$ denote a standardized telemetry vector for a single observation. The primary sketch is a random projection $y = Sx$ where $S \in \mathbb{R}^{m \times d}$ is drawn from a distribution that yields near-isometric embeddings for the relevant subset of vectors. A practical choice in streaming contexts is a sparse sign matrix where each column of S has a small number of nonzeros in $\{-1, +1\}$, enabling $O(\text{nnz}(x))$ update time for sparse x. The projected vector y is stored as a length-m float vector or quantized to reduce storage.

The role of y is to approximate Euclidean distances [20]. For two points x and x', the sketched distance is

$$\widehat{D}(x, x') = \|S(x - x')\|_2, \quad \text{and one expects} \quad (1 - \varepsilon)\|x - x'\|_2 \leq \widehat{D}(x \tag{2}$$

with high probability when m is sufficiently large relative to $\varepsilon^{-2} \log |\mathcal{W}|$. In telemetry, $|\mathcal{W}|$ can be large, but the effective intrinsic dimension may be smaller due to feature correlations and sparsity; nonetheless, m is chosen conservatively to avoid brittle behavior when new modes appear.

Because telemetry vectors may mix dense and sparse blocks, the sketch design benefits from block-aware scaling. Let $x = [x^{(c)}; x^{(s)}]$ where $x^{(c)}$ corresponds to continuous metrics and $x^{(s)}$ to sparse categorical or

LSH family	Tables L	Hashes/table K	Width / radius	Collision bias
Cosine LSH	64	8	N/A	Angular
p-stable (L2)	48	6	$w = 2.0$	Euclidean
Hamming bit-sampling	32	16	N/A	Bit flips
Hybrid (proposed)	72	10	$w = 1.5$	Mixed

Table 4: Locality-sensitive hashing parameter choices for different similarity measures.

Method	Upstream (MB/epoch)	Downstream (MB/epoch)	Total (MB/epoch)	Rel. reduction
Raw telemetry push	18,400	920	19,320	1.0×
Local-only outlier scoring	9,600	480	10,080	1.9×
Central sketching baseline	2,150	220	2,370	8.1×
Proposed sketch+LSH	890	140	1,030	18.8×

Table 5: Communication cost per training epoch compared to baseline approaches.

hashed features. Distances can be dominated by whichever block has larger variance, so we scale each block by robust estimates of dispersion. Write $\tilde{x}^{(c)} = \Gamma_c^{-1}(x^{(c)} - \mu_c)$ and $\tilde{x}^{(s)} = \Gamma_s^{-1} x^{(s)}$, where μ_c is a running robust location estimate and Γ_c, Γ_s are diagonal scale matrices. The combined standardized vector is $\tilde{x} = [\tilde{x}^{(c)}; \tilde{x}^{(s)}]$, and the primary sketch uses $S\tilde{x}$. To avoid shipping Γ at high resolution, we maintain sketches of scale statistics per feature family and exchange them periodically; the resulting alignment is approximate but sufficient for neighborhood ranking when the detector's threshold is tuned to an alert rate [21].

In addition to the projection sketch, we maintain auxiliary sketches that support normalization and robustness. A Count-Min style sketch of the absolute coordinate magnitudes of $x^{(s)}$ helps estimate the contribution of the sparse block and detect when a small set of categorical tokens dominates the representation. A compact norm sketch stores $\|\tilde{x}\|_2^2$ and optionally a clipped version to prevent a few extreme values from overwhelming distance. For cosine similarity variants, we store a normalized projection $\bar{y} = y / \max(\|y\|_2, \delta)$ where δ prevents division by very small norms for near-zero vectors.

Sketches must support sliding windows. Exact deletion is hard for many sketches, but in telemetry one can use time-partitioned sketches [22]. The window is divided into B buckets of width T/B, and each bucket maintains its own set of sketches and indices. When time advances, the oldest bucket is dropped entirely. This provides approximate sliding-window behavior with bounded memory and predictable deletion cost [23]. For per-point sketches used in neighbor scoring, time partitioning means the index stores sketches per point and discards points by bucket. For aggregate normalization sketches, each bucket stores its own moment sketches, and the current window is the sum of the active buckets.

Quantization is an important practical detail [24]. Floating point storage of y can be expensive at scale. We use stochastic or deterministic quantization to q-bit integers with per-bucket scale factors. Let $y \in \mathbb{R}^m$ and choose a scale $\alpha > 0$ such that $\|y\|_\infty \le \alpha$ for most points in a bucket. The quantized vector is $Q(y) \in \{-2^{q-1}, \dots, 2^{q-1} - 1\}^m$ with $Q(y)_j \approx y_j / \alpha$. Distance computations then use dequantized values $\alpha Q(y)$, introducing an additive noise term that can be bounded in expectation. In practice, the quantization error is dominated by sketch distortion when m is small, while for larger m quantization becomes the main contributor, so q is chosen jointly with m to balance compute, memory, and accuracy [25].

5 | Locality-Sensitive Hashing for Distributed Candidate Mining

The sketch layer makes approximate distance computation cheap once candidate neighbors are available, but it does not by itself solve the candidate generation problem. Searching over all points in $\mathcal{W}(t)$ remains infeasible. We therefore use LSH to generate candidate sets whose expected size is sublinear in the window size, while preserving a high probability of including true near neighbors.

We focus on two LSH families depending on the chosen metric [26]. For cosine similarity, we use random

Method	AUC-PR	AUC-ROC	F1@top-1%	Recall@top-0.1%
Isolation Forest	0.421	0.945	0.392	0.218
LOF (centralized)	0.456	0.952	0.407	0.236
Autoencoder (central)	0.489	0.961	0.432	0.261
GNN-based detector	0.517	0.968	0.451	0.278
Proposed distributed method	0.563	0.975	0.487	0.312

Table 6: Outlier detection quality on Cloud-Metrics telemetry using several baselines.

Workers	Latency / epoch (s)	Speedup vs 4 workers	Comm. cost (MB/epoch)	CPU util. (%)
4	412	1.0×	2,080	61
8	233	1.8×	1,520	67
16	131	3.1×	1,180	73
32	79	5.2×	1,040	76
64	63	6.5×	1,010	74

Table 7: Scalability of the method with respect to the number of worker nodes.

hyperplane hashing applied to a normalized projection. Let $r_\ell \in \mathbb{R}^m$ be random vectors with i.i.d. symmetric entries, and define bits

$$b_\ell(x) = \operatorname{sign}(r_\ell^\top \bar{y}), \quad \bar{y} = \frac{S\tilde{x}}{\max(\|S\tilde{x}\|_2, \delta)}. \tag{3}$$

Concatenating K such bits yields a hash key in $\{0,1\}^K$. The collision probability for two points depends on the angle between their normalized sketches, and monotonicity ensures that nearer points collide more often [27]. For Euclidean distance, we use p-stable LSH. Let $a \in \mathbb{R}^m$ have i.i.d. entries from a stable distribution appropriate for ℓ_2, and let b be uniform on $[0, r]$. The hash is [28]

$$h_{a,b}(x) = \left\lfloor \frac{a^\top y + b}{r} \right\rfloor, \quad y = S\tilde{x}. \tag{4}$$

Concatenating K such hashes yields a key in $\mathbb{Z}^K$. The bucket width r controls the trade-off between recall and candidate size.

In a centralized setting, one would build L hash tables, each with its own concatenation of K base hashes, and query all L tables to retrieve candidate points. In a distributed telemetry setting, the challenge is that points are stored across workers. Querying all workers per point is too expensive, and shipping complete hash tables is also expensive [29]. The proposed protocol instead exchanges bucket summaries. Each worker maintains local hash tables over its window buckets. Periodically, it sends to the coordinator a compact digest for each active hash table, consisting of bucket identifiers and small statistics [30]. The simplest digest is a Bloom-filter-like structure per hash table indicating which buckets are nonempty, but this can be too coarse under heavy load. A more informative digest stores a sample of bucket keys and approximate counts, optionally with a small reservoir sample of point identifiers per bucket.

The coordinator uses these digests to route queries only to workers that likely contain collisions [31]. When a new point x arrives at worker i, the worker computes its L keys and checks locally for candidates. In parallel, it sends the keys to the coordinator, which consults digests and returns a small set of worker IDs that have matching buckets. Only those workers are queried for candidate sketches in the relevant buckets, and responses include only sketches and lightweight metadata rather than raw vectors [32]. This design reduces the fanout from N to an expected number that depends on bucket occupancy and on the collision probabilities induced by the LSH parameters.

A key tuning problem is selecting (K, L, r) or the cosine analogue so that near neighbors collide with probability at least p_1 while far points collide with probability at most p_2, where $p_1 > p_2$. If one uses K-way concatenation, the per-table collision probabilities become p_1^K and p_2^K, and using L tables yields recall approximately $1 - (1 - p_1^K)^L$ while the expected number of far collisions scales with $L p_2^K$ times the number of points. In telemetry, the distance distribution can shift rapidly, so static p_1, p_2 targets may be unstable [33]. We therefore adapt K and L slowly based on observed bucket sizes and on the fraction of queries that return too few candidates for stable scoring. This adaptation uses only aggregate

Component	Count / node	Memory each (KB)	Total / node (MB)	Share of budget
Sketch instances	8	128	1.0	41%
LSH hash tables	64	32	2.0	48%
Routing metadata	1	128	0.1	4%
Buffering and queues	1	256	0.3	7%

Table 8: Memory footprint of the telemetry outlier detection stack at a typical node.

Sketch size (KB)	AUC-ROC	F1@top-1%	Rel. error vs 1,024 KB	Comm. cost (MB/epoch)
64	0.947	0.426	6.7%	780
128	0.957	0.447	4.1%	840
256	0.967	0.468	2.1%	910
512	0.972	0.480	1.2%	960
1,024	0.975	0.487	0.0%	1,030

Table 9: Effect of sketch size on accuracy and communication overhead for the proposed method.

statistics and does not require labels.

Because sketches already reduce dimension, it can be beneficial to apply LSH on the sketches rather than on raw vectors [34]. This is computationally cheaper and aligns the candidate generation metric with the scoring metric used downstream. However, sketch distortion can reduce the separation between near and far points, increasing p_2 and inflating candidate sets. The analysis section formalizes this coupling [35]. Practically, one can mitigate it by using a slightly larger m for the sketch used in hashing than for the sketch used in scoring, or by using two independent sketches, one optimized for hashing stability and one optimized for distance estimation. Independence reduces correlated errors where a point is both mis-hashed and mis-scored in the same direction.

6 Distributed Detection Protocol and Theoretical Analysis

The distributed protocol operates in micro-batches aligned to time buckets [36]. Each worker maintains active buckets for the current window and, within each bucket, stores per-point sketches and inserts point identifiers into L local hash tables. Periodically, each worker sends digests of its nonempty buckets to the coordinator. When processing a new point x, the worker computes its sketch y and its L hash keys. Local candidates are fetched from local buckets that match these keys [37]. Remote candidates are obtained by sending keys to the coordinator, receiving a small set of worker targets, and requesting from those workers the sketches of points in matching buckets, potentially limited by a cap that prevents worst-case blowups.

Given a candidate set $\mathcal{C}(x)$, the worker computes an approximate neighborhood score using only sketches. For a kNN-radius score, it estimates distances $\widehat{D}(x, z)$ for $z \in \mathcal{C}(x)$ and takes the kth order statistic. For an average-distance score, it takes the mean of the k smallest distances. A stabilized variant uses a trimmed mean to reduce sensitivity to occasional far candidates introduced by hash collisions [38]. Let $\widehat{d}_{(1)} \le \cdots \le \widehat{d}_{(|\mathcal{C}|)}$ be the sorted sketched distances. A generic score is

$$\widehat{s}(x) = \frac{1}{k} \sum_{j=1}^{k} \widehat{d}_{(j)} \quad \text{or} \quad \widehat{s}(x) = \widehat{d}_{(k)}, \tag{5}$$

and the outlier decision compares $\widehat{s}(x)$ to a threshold that is updated to match an alert rate budget. Thresholding uses robust quantile estimation over recent scores, implemented with mergeable quantile sketches to maintain a fleet-wide view without centralizing scores.

The analysis separates two error sources: candidate recall error and distance estimation error [39]. Candidate recall error occurs when true near neighbors are not included in $\mathcal{C}(x)$, causing the score to be biased upward and increasing false alerts. Distance estimation error occurs when $\widehat{D}$ differs from the true distance, potentially swapping neighbor order and affecting both false alerts and missed detections. Let $D(x, z) = \|x - z\|_2$ denote true distance in standardized space. Suppose the sketch preserves

distances within multiplicative distortion $(1 \pm \varepsilon)$ on the relevant subset of points. Then for any candidate z in that subset, [40]

$$(1-\varepsilon)D(x,z) \le \widehat{D}(x,z) \le (1+\varepsilon)D(x,z). \tag{6}$$

If $\mathcal{C}(x)$ contains the true k nearest neighbors, then the kNN radius computed on sketches satisfies

$$\widehat{d}_{(k)} \le (1+\varepsilon)\rho_k(x) \quad \text{and} \quad \widehat{d}_{(k)} \ge (1-\varepsilon)\rho_k(x), \tag{7}$$

up to the caveat that sketch distortion can change the ordering among neighbors with similar distances. The ordering sensitivity can be bounded in terms of the distance gap between the kth and $(k+1)$th neighbors, but telemetry often yields small gaps in dense regions, so multiplicative bounds on the score are more informative than exact neighbor recovery.

Candidate recall depends on LSH parameters and on the separation between near and far points in the sketched metric. Let $z^\star$ be a true near neighbor of x with $D(x,z^\star) \le R$, and let z be a far point with $D(x,z) \ge cR$ for some $c > 1$ [41]. Under an LSH family, define per-table collision probabilities p_1 for near pairs and p_2 for far pairs after concatenation of K base hashes. With L tables, the probability that $z^\star$ is missed by all tables is $(1-p_1)^L$, so recall is at least $1-(1-p_1)^L$. The expected number of far candidates returned is proportional to Lp_2 times the number of far points in the window, though in practice bucket capping and digest routing reduce this [42]. The coupling to sketches enters because the LSH is applied to $y = S\tilde{x}$ rather than to $\tilde{x}$. If the sketch distorts distances, then a pair that is near in the original space may appear farther in the sketched space, reducing p_1, while some far pairs may appear nearer, increasing p_2.

A simplified bound can be derived by assuming that the LSH collision probability is Lipschitz in the underlying distance parameter over the relevant range. Let $p(D)$ denote the collision probability for a pair at distance D in the metric used by LSH [43]. If p is differentiable and $|p'(D)| \le L_p$ on $[R, (1+\varepsilon)cR]$, then sketch distortion yields

$$p\big(\widehat{D}(x,z^\star)\big) \ge p(R) - L_p\varepsilon R, \quad p\big(\widehat{D}(x,z)\big) \le p(cR) + L_p\varepsilon cR \tag{8}$$

and these translate to perturbed p_1, p_2 for concatenated hashes. Although this bound is loose, it highlights that increasing m to reduce ε can improve both scoring accuracy and candidate quality simultaneously, while quantization and normalization errors can have the opposite effect [44]. In telemetry, where distance scales can shift across time, it is useful to treat R as a quantile of observed neighbor distances and update it online. This results in an adaptive operating point where LSH parameters are tuned to the current regime rather than to a fixed offline estimate.

Communication complexity depends on digest size, query routing fanout, and candidate sketch transfers. Let U be the number of active buckets per worker, and let $B_{i,\ell}$ be the number of nonempty buckets in hash table ℓ at worker i. If each digest entry includes a bucket key and an approximate count, digest communication per epoch scales with $\sum_{i,\ell} B_{i,\ell}$. Candidate transfers dominate when many collisions occur [45]. Suppose each query results in F remote workers being contacted on average and each contact returns C candidate sketches of size m with q-bit quantization. Then the per-query bandwidth is roughly $FCmq$ bits plus overhead. The system controls this by enforcing caps on C, by increasing K when buckets become too large, and by using multi-probe strategies that query nearby buckets when recall is low rather than decreasing K globally [46]. Because telemetry often has bursty periods, the caps prevent pathological worst-case costs at the expense of a small, controlled reduction in recall.

7 | Experimental Methodology and Discussion

Evaluation in distributed telemetry is complicated by the absence of definitive labels and by the heterogeneity of normal behavior across services. The methodology used here emphasizes comparative behavior under controlled stressors and consistency with known operational heuristics [47]. Data are constructed from mixtures of continuous metrics and categorical tags, where continuous features follow heavy-tailed and correlated distributions, and categorical features are generated from a Zipf-like frequency law to mimic the long tail of tag values. To mimic drift, the generator changes feature means and variances over time and introduces new categorical values. To mimic incidents, the generator injects rare patterns that combine coordinated changes across a subset of features with the appearance of new tag combinations [48]. The injected patterns are designed so that they are not trivially detectable by univariate thresholds, thereby testing whether neighborhood structure provides added value.

Baselines include centralized kNN scoring on raw standardized vectors, local-only scoring that ignores cross-worker neighbors, and distributed LSH without sketches where candidate points are fetched as raw

vectors. The proposed method is evaluated under varying window sizes, sketch dimensions, quantization levels, and LSH parameters. Metrics include rank correlation of outlier scores relative to the centralized baseline, precision at a fixed alert rate such as 0.1% or 1%, recall of injected incidents, and communication measured as bytes per point and bytes per second [49]. Latency is measured as time from point arrival to decision under a simulated network with variable delay and loss.

Across regimes, several qualitative patterns emerge. When drift is mild and the window is large, the centralized baseline yields stable neighborhoods and the sketch-only method closely tracks it for moderate m, with most score discrepancies occurring in dense regions where many points have similar neighbor distances [50]. In such regions, outlier decisions are less sensitive because the score distribution is narrow, and the alert threshold can be adjusted to maintain the target rate. When drift is rapid, the benefits of global neighborhoods diminish because points become incomparable across time, and windowing dominates performance. In this setting, the sketch-only method behaves similarly to centralized kNN if both use the same windowing, but both can fail to separate drift from incidents unless normalization is drift-aware [51]. The protocol's bucketed windowing helps because it naturally weights recent buckets more heavily if one uses an exponentially decaying aggregation of normalization sketches.

Communication reductions are largest when the fleet is large and the per-worker load is moderate, since candidate routing fanout remains small due to bucket sparsity. In contrast, when many workers observe highly similar telemetry, such as during a fleet-wide rollout where metrics align, LSH buckets become dense and candidate caps activate frequently [52]. This increases the variance of scores because candidate sets become truncated. The adaptive parameter tuning mitigates this by increasing K to split buckets, but doing so can reduce recall for borderline neighbors. A practical compromise is to maintain two LSH configurations simultaneously, one coarse for high recall and one fine for high precision, and to fuse their candidate sets subject to a combined cap. Even without explicitly maintaining two configurations, one can emulate this by using multi-probe querying on a fine configuration to recover some recall [53].

Quantization has a measurable but controllable effect. With aggressive quantization, distance estimates become noisier and neighbor ordering becomes less stable, especially for small m. Increasing m partially compensates because averaging over more projected dimensions reduces sensitivity to quantization noise, but this increases both storage and bandwidth per candidate sketch [54]. Empirically, mid-range quantization yields a favorable trade-off, where score rank correlation remains high while per-candidate payload remains small. The remaining errors tend to concentrate in points with very small norms after standardization, where relative error is magnified. The protocol addresses this by enforcing a minimum norm floor δ in normalization and by treating near-zero vectors as a separate regime with a specialized similarity measure [55].

Finally, the distributed aspects introduce their own distortions. When digests are delayed, the coordinator's routing may omit relevant workers, reducing candidate recall. The impact depends on the window width and on whether incidents are localized or global [56]. If incidents are localized to a subset of workers, missing those workers reduces recall substantially. If incidents are global, local candidates often suffice. A mitigation is to incorporate a small amount of randomized probing, where a query is occasionally sent to a few random workers regardless of digest matches. This increases bandwidth slightly but reduces worst-case blind spots under digest staleness [57]. Another mitigation is to increase digest update frequency during detected bursts, which can be triggered by observing sudden changes in bucket occupancy distributions.

8 | Conclusion

This paper presented a distributed outlier detection method for high-dimensional telemetry that combines sketch-based representations with locality-sensitive hashing for candidate neighborhood generation. The method is motivated by the operational constraints of telemetry processing, where full centralization and exact neighbor search are often infeasible [58]. Linear sketches provide compact, streaming-friendly representations that enable approximate distance computations and mergeable normalization statistics, while LSH yields a principled mechanism for reducing the candidate search space in high dimension. The distributed protocol exchanges bucket-level digests and candidate sketches rather than raw vectors, enabling cross-worker neighborhood scoring with substantially reduced communication.

The analysis clarified how sketch distortion and quantization interact with LSH collision probabilities and how these effects propagate to neighborhood-based outlier scores [59]. While the bounds are conservative, they identify the primary trade-offs that govern

practical parameter selection, including sketch dimension, bucket width, concatenation length, and number of hash tables. The discussion of experimental behavior highlighted regimes where the method closely tracks centralized baselines and regimes where drift, dense buckets, or stale routing information can degrade performance. The resulting picture is that sketches and LSH together can support scalable, bandwidth-aware outlier detection when coupled with drift-aware normalization, adaptive parameter tuning, and safeguards such as candidate caps and occasional randomized probing [60].

Several extensions are natural within the same framework. More expressive sketches could support kernelized similarities that better capture sparse categorical interactions, and adaptive feature weighting could reduce the impact of noisy or unstable telemetry dimensions. The distributed protocol could be generalized to hierarchical coordinators or to fully peer-to-peer routing to reduce single points of congestion. These directions retain the central principle developed here: preserving enough neighborhood structure to score outliers while treating communication as a first-class constraint [61].

References

[1] R. Lichtenthäler and G. Wirtz, "An experimental validation of architectural measures for cloud-native quality evaluations," in *2025 IEEE 18th International Conference on Cloud Computing (CLOUD)*, pp. 374–384, IEEE, 7 2025.

[2] D. Woos, Z. Tatlock, M. D. Ernst, and T. Anderson, "A graphical interactive debugger for distributed systems.," 6 2018.

[3] L. Ciuffreda, "Distributed systems for neural network models," 10 2018.

[4] N. Naik, "Isse - demystifying properties of distributed systems," in *2021 IEEE International Symposium on Systems Engineering (ISSE)*, pp. 1–8, IEEE, 9 2021.

[5] S. Bagchi, "Modeling slicer and control algorithm for distributed computation in metrized weaker topological spaces," *International Journal of Control and Automation*, vol. 10, pp. 209–220, 12 2017.

[6] C. Tang, Z. Wang, X. Zhang, Q. Yu, B. Zang, H. Guan, and H. Chen, "Many faces of ad hoc transactions," *Communications of the ACM*, vol. 68, pp. 71–80, 3 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] "Proceedings of the 1st international symposium on parallel computing and distributed systems," in *2024 International Symposium on Parallel Computing and Distributed Systems (PCDS)*, pp. 1–1, IEEE, 9 2024.

[9] N. B. Bhat, D. Madurasinghe, I. Ozcelik, R. R. Brooks, G. K. Venayagamoorthy, and A. Skjellum, *Evaluation and Design of Performable Distributed Systems*, pp. 211–227. Springer International Publishing, 11 2020.

[10] 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.

[11] S. Dustdar, "Keynote: Engineering the new fabric of the distributed compute continuum," in *2022 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 65–65, IEEE, 3 2022.

[12] S. Devismes, "Versatility and efficiency in self-stabilizing distributed systems," 12 2020.

[13] R. K. Mishra and S. R. Raman, *Introduction to PySpark SQL*, pp. 1–22. Apress, 3 2019.

[14] 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.

[15] S. Kleber and F. Kargl, "Refining network message segmentation with principal component analysis," in *2022 IEEE Conference on Communications and Network Security (CNS)*, pp. 281–289, IEEE, 10 2022.

[16] R. Rotta, J. Schulz, B. Naumann, N. S. Chatharajupalli, J. Nolte, and M. Werner, "B.a.t.m.a.n. mesh networking on esp32's 802.11," in *2024 IEEE 49th Conference on Local Computer Networks (LCN)*, vol. 3, pp. 1–7, IEEE, 10 2024.

[17] A. Tello and V. Degeler, "Digital twins: An enabler for digital transformation," 6 2021.

[18] A. Lackinger, P. A. Frangoudis, I. Čilić, A. Furutanpey, I. Murturi, I. P. Žarko, and S. Dustdar, "Inference load-aware orchestration for hierarchical federated learning," in *2024 IEEE 49th Conference on Local Computer Networks (LCN)*, pp. 1–9, IEEE, 10 2024.

[19] F. Lu, X. Wei, Z. Huang, R. Chen, M. Wu, and H. Chen, "Serialization/deserialization-free state transfer in serverless workflows," in *Proceedings of the Nineteenth European Conference on Computer Systems*, pp. 132–147, ACM, 4 2024.

[20] 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.

[21] C.-F. Cheng and C.-W. Huang, "The harmonized consensus protocol in distributed systems," *The Journal of Supercomputing*, vol. 75, pp. 7690–7722, 8 2019.

[22] A. Fieschi, P. Hirmer, S. Agrawal, C. Stach, and B. Mitschang, "Hysaad – a hybrid selection approach for anonymization by design in the automotive domain," in *2024 25th IEEE International Conference on Mobile Data Management (MDM)*, pp. 203–210, IEEE, 6 2024.

[23] J. Tournier, F. Lesueur, F. L. Mouël, L. Guyon, and H. Ben-Hassine, "Iotmap, a modelling system for heterogeneous iot networks," 6 2020.

[24] O. Scekic, S. Nastic, and S. Dustdar, "Blockchain-supported smart city platform for social value co-creation and exchange," *IEEE Internet Computing*, vol. 23, pp. 19–28, 1 2019.

[25] R. Kuznets, "Communication modalities," 5 2024.

[26] T. Simeonova, "Peculiarities of distributed systems, scada and iot," *Yearbook Telecommunications*, vol. 7, pp. 65–70, 8 2021.

[27] A. A. T. Thulnoon, "Efficient runtime security system for decentralised distributed systems," 8 2018.

[28] S. Dahdal, F. Poltronieri, A. Gilli, M. Tortonesi, R. Fronteddu, R. Galliera, and N. Suri, "Roamml: Distributed machine learning at the tactical edge," in *MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM)*, pp. 33–38, IEEE, 10 2023.

[29] M. ter Beek, B. Re, M. Viroli, R. Anane, and R. Bahsoon, "Session details: Distributed systems: Ccs track," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, ACM, 4 2018.

[30] N. Bronson, A. Aghayev, A. Charapko, and T. Zhu, *HotOS - Metastable failures in distributed systems*. ACM, 6 2021.

[31] K. Cui, S. Liu, W. Feng, X. Deng, L. Gao, M. Cheng, H. Lu, and L. T. Yang, "Correlation-aware cross-modal attention network for fashion compatibility modeling in ugc systems," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 21, pp. 1–24, 11 2025.

[32] 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.

[33] W. B. Daszczuk, *Fairness in Distributed Systems Verification*, pp. 139–159. Germany: Springer International Publishing, 3 2019.

[34] J. Breuer, B. Čemusová, J. Fischer, J. Roztočil, and V. Vigner, *Synchronization of Distributed Systems using GPS*, pp. 95–120. River Publishers, 10 2024.

[35] X. Wei, F. Lu, Z. Huang, R. Chen, M. Wu, and H. Chen, "Towards serialization/deserialization-free state transfer in serverless workflows," *ACM Transactions on Computer Systems*, vol. 43, pp. 1–32, 11 2025.

[36] "Evaluation of safety and survivability of distributed system using complexity proportionality assessment (copras) method," *Journal on Innovations in Teaching and Learning*, vol. 4, pp. 24–33, 9 2025.

[37] I. Colonnelli and M. Aldinucci, "Workflow models for heterogeneous distributed systems," 5 2022.

[38] *ACSD - Concurrent Secrets with Quantified Suspicion*, 4 2018.

[39] J. Stój, A. Ziebinski, and R. Cupek, "Fpga based industrial ethernet network analyser for real-time systems providing openness for industry 4.0," *Enterprise Information Systems*, pp. 1–21, 7 2021.

[40] C. Gao, Y. Zhong, and X. He, *Safety Control of Distributed Systems*, pp. 153–159. Elsevier, 1 2024.

[41] M. Maresch and S. Nastic, "Vate: Edge-cloud system for object detection in real-time video streams," in *2024 IEEE 8th International Conference on Fog and Edge Computing (ICFEC)*, pp. 27–34, IEEE, 5 2024.

[42] M. Anisetti, C. A. Ardagna, E. Damiani, and N. E. Ioini, *Certification of Modern Distributed Systems*, pp. 41–60. Springer International Publishing, 7 2024.

[43] C. Tang, Z. Wang, J. Li, and H. Chen, "Sonata: Multi-database transactions made fast and serializable," *Proceedings of the VLDB Endowment*, vol. 18, pp. 3449–3462, 9 2025.

[44] D. Efimov, A. Polyakov, and A. Aleksandrov, "Proofs for "discretization of homogeneous systems using euler method with a state-dependent step"," 6 2019.

[45] 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.

[46] 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.

[47] *Convergence conditions for Persidskii systems*, 6 2021.

[48] Z. Lu, P. Xu, Y. Wang, Y. Yang, Q. Chen, W. Yu, and G. Qu, "An fpga-based key-switching accelerator with ultra-high throughput for fhe," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, ACM, 10 2024.

[49] Y. Teng, J. Qi, L. Liu, S. Wang, L. Xu, and C. Fan, "Secure synchronized spatio-temporal trajectory similarity search," in *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 964–973, IEEE, 11 2023.

[50] P. Waibel, C. Hochreiner, S. Schulte, A. Koschmider, and J. Mendling, "Viepep-c: A container-based elastic process platform," *IEEE Transactions on Cloud Computing*, vol. 9, pp. 1657–1674, 10 2021.

[51] S. Dovgyi, O. Trofymchuk, O. Lebid, I. Kaliukh, V. Berchun, and Y. Berchun, "Aeroelastic flutter oscilations of distributed systems," in *2022 IEEE 3rd KhPI Week on Advanced Technology (KhPIWeek)*, pp. 1–5, IEEE, 10 2022.

[52] I. Colonnelli and M. Aldinucci, "Workflow models for heterogeneous distributed systems," 5 2022.

[53] R. Achar, P. Dawn, and C. V. Lopes, "Gotcha: An interactive debugger for got-based distributed systems.," 9 2019.

[54] S. Sakr and A. Y. Zomaya, *Distributed Systems*, pp. 690–690. Springer International Publishing, 2 2019.

[55] S. Shetty, C. A. Kamhoua, and L. Njilla, "Blockchain for distributed systems security," 2 2020.

[56] T. M. Tatarnikova and E. M. Arkhiptsev, "Hybrid time synchronization in distributed systems," in *2025 XXVIII International Conference on Soft Computing and Measurements (SCM)*, pp. 307–310, IEEE, 5 2025.

[57] A. Kathait and S. N. Dhage, "Fractal load balancing method in distributed systems," in *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, pp. 172–176, IEEE, 10 2020.

[58] 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.

[59] 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.

[60] A. Moumen, L. Zahiri, M. Jammoukh, and K. Mansouri, *Numerical Modeling of the Thermomechanical Behavior of Polypropylene Reinforced by Snail Shell Particles as a Sustainable and Ecological Biocomposite*,

pp. 359–369. Springer International Publishing, 2 2022.

[61] E. Börger and A. Raschke, *Modeling Distributed Systems*, pp. 207–239. Springer Berlin Heidelberg, 4 2018.