Scientific Journals
An international open-access platform

Hanlin Letters

Article

OPEN ACCESS

Approximate Query Processing over Data Lakes Using Semantic Sampling and Vector-Similarity Guided Stratification for E-Commerce Marketplace BI

Tran Hoang Phuc,[1] Vo Minh Khang[2]

[1] Department of Information Systems, Saigon Institute of Technology and Innovation, Đường 30 Tháng 4, Phường Phú Thọ, Thủ Dầu Một, Vietnam;
[2] Department of Software Engineering, Central Highlands University, Đường Trần Hưng Đạo 102, Phường Tân Thạnh, Tam Kỳ, Vietnam;

Abstract

E-commerce marketplaces increasingly rely on data lakes that consolidate event streams, transactional records, product catalogs, seller operations, logistics telemetry, and customer interactions. Business intelligence workloads over such lakes are dominated by exploratory aggregates, repeated slice-and-dice filtering, and wide group-by summaries whose exact evaluation can be expensive under high data volume, heterogeneous formats, and evolving schemas. Approximate query processing offers a practical path to interactive latency by trading small, controlled error for substantial reductions in scanned data, but classical sampling and stratification strategies often underperform in marketplaces because relevant subpopulations are defined by high-cardinality, semantically rich attributes such as textual product descriptions, user intent signals, and behavioral embeddings rather than by stable, low-dimensional keys. This paper develops an approximate query processing framework for data-lake-based marketplace BI that combines semantic sampling with vector-similarity guided stratification. The central idea is to represent tuples, entities, and query predicates in a shared embedding space, maintain semantic samples that preserve rare but decision-critical regions, and at query time allocate sampling effort toward strata whose semantic proximity to the query indicates higher contribution and variance. The approach yields unbiased or asymptotically unbiased estimators for common aggregates, supports group-by over dynamic taxonomies, and provides error estimation that remains informative under distribution shift. The proposed design integrates with lakehouse storage, catalog metadata, and feature stores while supporting incremental maintenance. Analytical modeling and empirical considerations highlight conditions under which semantic stratification reduces variance relative to key-based baselines, particularly for long-tail products, sparse conversion events, and cross-domain joins.

1. Introduction

E-commerce marketplace business intelligence operates in an environment where the underlying data is simultaneously large, diverse, and rapidly changing [1]. A single marketplace produces clickstream events, search queries, add-to-cart events, checkout funnels, payment authorizations, returns and refunds, customer support interactions, seller listing updates, promotions, and logistics scans. These data sources arrive with different latencies, different quality guarantees, and different structures, ranging from strongly typed transactional tables to semi-structured JSON events and unstructured text. Data lakes provide a flexible substrate for storing and unifying such sources, but flexibility comes with computational cost. Analysts and automated reporting systems must repeatedly execute aggregates and group-bys across wide tables and joins, frequently filtered by a mixture of structured predicates and semantic concepts such as product similarity, intent similarity, fraud risk, or customer segment embeddings. Even in lakehouse architectures with columnar storage and indexing, exact evaluation can remain expensive because relevant signals are scattered across multiple domains and because the queries that matter for decision making often

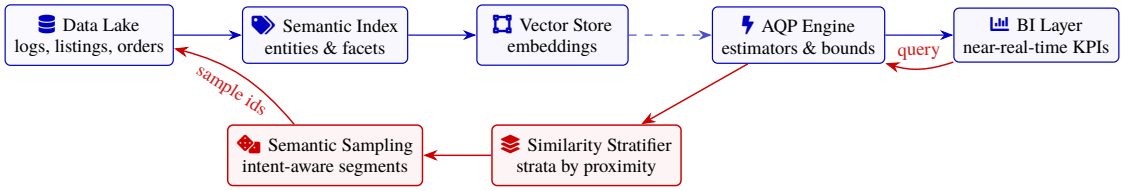

Figure 1: System overview for marketplace BI: approximate query execution is driven by intent-aware semantic sampling and vector-similarity guided stratification over a heterogeneous data lake, producing fast KPI estimates with controlled uncertainty.

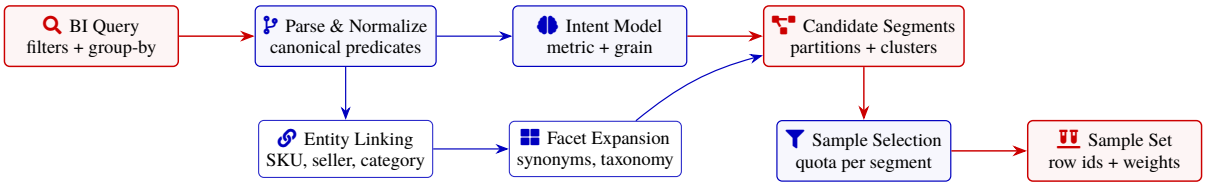

Figure 2: Semantic sampling pipeline: normalized BI queries are mapped to business intent and linked entities, then expanded through marketplace taxonomies to produce candidate segments and a weighted sample set for approximate aggregation.

focus on sparse events in the long tail rather than on the head of the distribution [2].

Approximate query processing (AQP) addresses these constraints by producing answers with quantifiable error using partial data access. For BI use cases, latency reductions matter not only for interactive exploration but also for operational dashboards that must refresh frequently, for anomaly detection that scans many slices, and for experimentation pipelines where analysts iterate through variants of the same metric definition. In marketplaces, however, classical AQP is challenged by a combination of high cardinality and semantic heterogeneity. Consider measuring conversion rate by product category and seller region while filtering on an intent cluster derived from search queries, or estimating the uplift of a promotion for semantically similar products to a seed item. The subpopulation relevant to the query is not well described by a small set of stable keys [3]. Instead, relevance is encoded in text, images, user behavior sequences, and learned embeddings that compress these modalities into vectors. If sampling ignores this semantic structure, the resulting samples can miss rare but important regions, yield unstable estimates for tail segments, and misrepresent the conditional distributions induced by semantic filters.

The tension is particularly clear in marketplace BI because decision making routinely depends on tail phenomena. Fraud signals are sparse, policy violations are rare, and certain operational incidents are localized. Even standard growth metrics can be sensitive to segments that are small in volume but large in revenue, such as high-value customers or niche product clusters [4]. AQP systems that rely on uniform sampling or simple stratification by a few categorical attributes tend to concentrate resolution where data volume is high, leaving tail segments with high relative error. Conversely, maintaining per-segment samples for all combinations is infeasible because segment definitions change and because the number of possible group-by keys explodes. The challenge is to find a representation and sampling strategy that generalizes across segment definitions while preserving fidelity for semantically defined subsets.

Learned vector representations provide a promising handle. Modern marketplace stacks already maintain embeddings for products, queries, users, and sessions to support ranking, retrieval, recommendations, and personalization. These embeddings encode semantic proximity in a continuous space where similarity can proxy for shared attributes that are not explicitly modeled in schemas [5]. Embeddings also support generalization: a new product can be embedded near similar products even if its categorical attributes are missing or inconsistent, and a new query can be embedded near known intents even if its surface form is novel. For BI, this suggests that sampling and stratification could be guided by embeddings to preserve the structure of semantically defined subpopulations and to adapt sampling effort to the parts of the space most relevant to a given query.

This paper proposes an AQP framework for data-lake-based marketplace BI that combines semantic sampling and vector-similarity guided stratification. Semantic sampling refers to maintaining samples that are not merely random subsets but are constructed to preserve distributional properties in embedding space, including tail regions and semantically coherent neighborhoods. Vector-

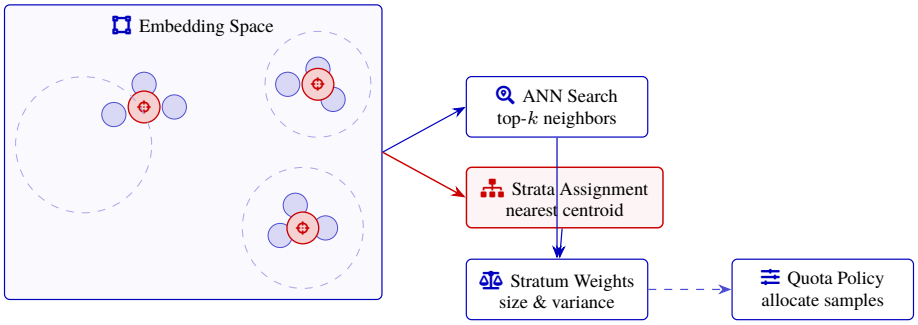

Figure 3: Vector-similarity guided stratification: marketplace records are embedded, grouped around learned centroids, and assigned to strata using nearest-neighbor proximity; stratum-level weights and quotas steer sampling toward segments with higher impact on estimator variance.

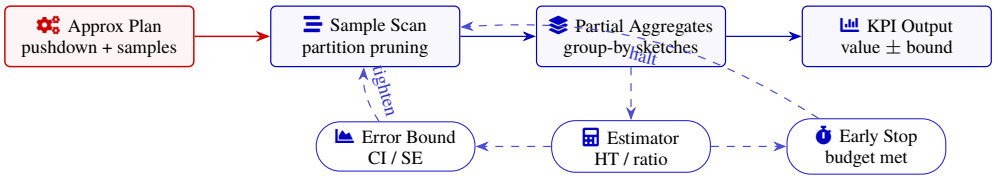

Figure 4: Approximate execution path: a sample-aware plan scans pruned partitions, forms partial aggregates, and applies unbiased estimators with uncertainty bounds; stopping criteria trade accuracy for latency under BI budgets.

similarity guided stratification refers to creating or selecting strata based on embedding clusters and, at query time, allocating sampling budget across strata using similarity between the query representation and stratum representations [6]. The goal is to reduce estimator variance for semantically filtered aggregates while keeping sample maintenance tractable and ensuring that error estimates remain meaningful under drift.

The approach is designed to fit data lake realities. Data is stored in partitioned columnar formats, catalogs provide schemas and lineage, and compute is distributed. Queries may join facts to dimensions, combine event data with catalog metadata, and aggregate across time windows. Semantic representations may live in a feature store and be referenced via keys [7]. The framework treats embedding functions as part of the logical metadata, enabling the AQP layer to reason about semantic similarity even when raw data is heterogeneous. The sampling layer can be maintained incrementally and can leverage compaction and clustering operations already present in lakehouse systems.

A key requirement for marketplace BI is that approximate answers must be interpretable and stable. Analysts accept approximation when the error bounds are credible and when the system behaves predictably across repeated runs. Therefore, the paper emphasizes estimators with well-understood properties, online variance estimation, and mechanisms for diagnosing when approximation is unsafe, such as under severe covariate shift or when queries target extremely small populations. The framework also accounts for the fact that many BI queries are parameterized templates executed repeatedly, which enables amortizing the cost of building semantic samples and updating strata [8].

The remainder of the paper develops the technical components of the framework. The next section formalizes the problem setting in the context of data lakes and marketplace BI, introduces embedding-aware representations, and defines the query classes of interest. The following sections present semantic sampling strategies for building and maintaining sample indexes, then introduce vector-similarity guided stratification and budget allocation. Subsequent modeling develops estimators and error quantification for aggregates and group-bys under semantic stratification, including join-aware considerations. The final technical section discusses system integration in a lakehouse environment and examines practical trade-offs, including drift handling and resource governance, before concluding with implications and limitations [9].

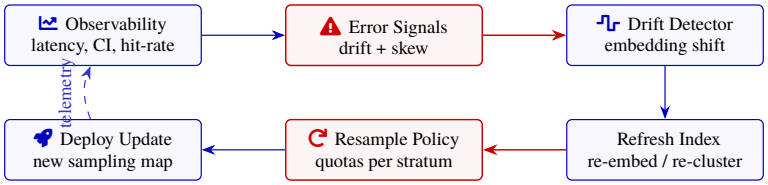

Figure 5: Adaptive loop for production BI: telemetry and error signals drive drift detection, triggering index refresh and quota updates so that semantic samples remain representative as marketplace behavior and catalog content evolve.

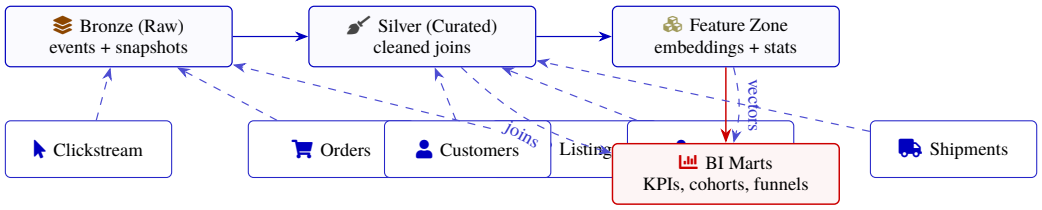

Figure 6: Data-lake organization for marketplace analytics: raw behavioral and transactional sources are curated into conformed entities, enriched with vector features, and served into BI marts that support stratified sampling across high-variance segments (e.g., category, seller tier, traffic channel).

2. Problem Setting and Semantic Representation in Data Lakes

A data lake for a marketplace can be modeled as a collection of datasets with heterogeneous schemas and formats. Let the lake contain relations $R_1, R_2, \ldots, R_m$, where each relation may represent a fact table such as impressions or orders, a dimension table such as product metadata, or a log-derived table such as search sessions. Each relation R consists of tuples $x \in \mathcal{X}_R$ with attributes that may include numeric measures, categorical identifiers, timestamps, and text fields. Queries for BI often compute aggregates over time windows and conditional filters, frequently joining facts to dimensions. A generic query can be viewed as producing an aggregate of a measure function $g(x)$ over tuples that satisfy a predicate $P(x)$, optionally grouped by keys $h(x)$. In exact execution, the system scans the relevant partitions, applies joins and filters, and computes aggregates [10].

AQP aims to approximate such queries by evaluating them on a subset of tuples and scaling appropriately. Classical sampling assumes that the predicate P is expressible over structured fields and that the distribution induced by P is not too skewed relative to the sampling design. In marketplace BI, predicates frequently include semantic constraints, either explicitly, such as filtering products similar to a seed item, or implicitly, such as filtering by an intent cluster or risk score derived from a model. Even when predicates are expressed as structured filters, the relevant subpopulations are often semantically correlated with unstructured fields, leading to hidden stratification that uniform sampling does not capture. For example, a filter on a set of category identifiers may correspond to multiple semantically distinct product clusters with different conversion behaviors, and a filter on seller region may interact with product semantics and logistics constraints [11].

To capture these interactions, the framework assumes the existence of embedding functions that map tuples or entities into a vector space. For a tuple x, let $\phi(x) \in \mathbb{R}^d$ denote an embedding that summarizes the semantics relevant to BI queries. In practice, ϕ may be composed from multiple embeddings, such as a product embedding derived from title and image, a user embedding derived from behavior, and a context embedding derived from query intent, concatenated or projected into a shared space. The AQP layer does not require a specific model class, but it requires that the embeddings satisfy a stability property: semantically similar entities map to nearby vectors, and the mapping is sufficiently smooth that neighborhoods correspond to coherent subpopulations.

Queries that involve semantic concepts can also be represented in the same space. A predicate P that selects items similar to a seed s can be represented by a query vector $q = \phi(s)$ and a similarity threshold. More general predicates that involve multiple conditions can be embedded by a function $\psi(P)$ that maps the predicate into $\mathbb{R}^d$, using, for example, a text encoder over the filter description or a learned mapping from structured constraints to vectors. The representation is not assumed to be perfect;

Collection	# Files (M)	Size (TB)	Domain Coverage
Product Events	2.1	120	Clicks, views, cart actions
Transaction Logs	0.4	65	Orders, payments, refunds
Search Interactions	1.7	48	Queries, impressions, CTR
Catalog Snapshots	0.3	31	Listings, attributes, prices
Recommendation Traces	0.9	22	Served slots, feedback

Table 1: Data lake collections used in the e-commerce marketplace experiments.

Embedding Model	Dimensionality	Tokenizer	Pretraining Source
prod-text-mini	128	BPE	Product titles & categories
marketplace-sem-base	256	WordPiece	Titles, queries, reviews
search-intent-large	384	SentencePiece	Query–click pairs
hybrid-multi-modal	512	Dual-stream	Text + image embeddings

Table 2: Semantic embedding models used for vector representation of data lake records.

rather, it is treated as a noisy but useful signal for allocating sampling effort [12].

The primary query class considered is aggregate queries with optional group-by, potentially involving joins between a large fact relation and smaller dimension relations. Let F denote the fact relation with tuples x, and let D denote a set of dimension relations. A joined tuple is $z = x \bowtie d$ where d aggregates the dimension attributes. The measure function $g(z)$ can represent revenue, units, gross margin, conversion indicators, latency, or risk. The predicate $P(z)$ can include time constraints, structured filters, and semantic similarity filters [13]. The goal is to estimate aggregates such as

$$A = \sum_{z \in F \bowtie D} g(z)\,\mathbf{1}\{P(z)\}, \quad \text{or} \quad \mu = \frac{\sum_z g(z)\,\mathbf{1}\{P(z)\}}{\sum_z \mathbf{1}\{P(z)\}}. \tag{1}$$

For group-by queries, the aggregate is computed per group key $y = h(z)$, producing A_y or μ_y. In dashboards, analysts often care about a subset of groups, such as top sellers, key categories, or anomalous regions, but exploratory workflows may request many groups at once.

The data lake setting imposes constraints on sampling. Data is partitioned, often by time and sometimes by high-level keys [14]. Maintaining a global sample must respect partition boundaries to avoid expensive reshuffles. Updates are continuous, so samples must be maintainable incrementally. Governance requires that sampling not leak sensitive information and that it integrate with access control. Finally, semantic embeddings may evolve as models are retrained, which creates drift in the embedding space. An AQP design that relies on fixed clusters in embedding space must therefore include a strategy for handling embedding drift without full recomputation.

The framework addresses these constraints by maintaining semantic sample indexes that are partition-aligned and embedding-aware, and by using vector-similarity guided stratification that can adapt to query vectors even when embeddings drift [15]. Semantic samples are designed to preserve representation across embedding neighborhoods and across rare strata defined by business-critical outcomes. The stratification layer uses embedding clusters and metadata-derived strata, and it allocates query-time sampling effort based on similarity between the query vector and stratum centroids or density estimates. The design thus separates offline maintenance, which can be scheduled and governed, from online query-time allocation, which must be fast and predictable.

An important subtlety is that many marketplace BI metrics depend on joins where the join keys are stable but the semantics are not. Product embeddings live on the product dimension, while events live on fact tables [16]. The sampling design must ensure that when events are sampled, the distribution of product embeddings among sampled events remains representative of the join-induced population under relevant predicates. This motivates join-aware sampling where the sampling probability can depend on the embedding of the joined dimension entity, even if the sample is stored with the fact relation. The next section develops semantic sampling strategies that implement these ideas while keeping maintenance feasible in a data lake.

Strategy	Strata Definition	Similarity Signal	Complexity
Random Uniform	None	None	$O(N)$ sampling
Schema-driven	Partition + columns	None	$O(N \log K)$
Vector-kMeans	Embedding clusters	Cosine	$O(ITNK)$
Hybrid Semantic	Schema + clusters	Cosine + Jaccard	$O(ITNK)$

Table 3: Stratification strategies compared in the approximate query processing framework.

Method	Rel. Error @p95 (%)	Latency (s)	Speedup vs Exact
Exact Scan (Presto)	0.0	41.8	1×
Random AQP	12.4	4.9	8.5×
Schema-stratified AQP	9.7	5.3	7.9×
Vector-stratified AQP	6.1	4.1	10.2×
Semantic + Vector AQP	3.8	4.4	9.5×
Learned Synopsis AQP	5.5	3.7	11.3×

Table 4: Latency–accuracy trade-offs for approximate query processing methods over the BI workload.

3. Semantic Sampling: Embedding-Aware Sample Maintenance

Semantic sampling aims to maintain samples that preserve structure in embedding space so that approximate aggregates conditioned on semantic predicates have low variance. A straightforward approach would be to cluster all tuples by embeddings and maintain uniform samples within each cluster [17]. In a data lake, this is expensive because tuples arrive continuously, embeddings may be computed asynchronously, and the clustering must be updated under drift. Moreover, marketplace data exhibits multi-scale structure: some regions of embedding space are dense and stable, while others correspond to rapidly changing trends, new products, or adversarial behavior. A semantic sampling design must therefore be incremental, robust, and sensitive to tail importance.

The framework models sampling as assigning each tuple z an inclusion probability $\pi(z)$, potentially dependent on its embedding $\phi(z)$, its time partition, and auxiliary signals such as a measure proxy. The sample is then drawn either as a Bernoulli sample or via reservoir-style selection per partition [18]. For unbiased estimation of sums, it suffices that $\pi(z)$ be known or estimable, enabling Horvitz–Thompson style estimators. In practice, exact knowledge of $\pi(z)$ is feasible if sampling decisions are made deterministically from a hash and a target rate within strata. Semantic sampling thus focuses on defining strata and rates that are embedding-aware.

A core idea is to define a semantic partition of embedding space into regions $C_1, \ldots, C_K$, where each region corresponds to a cluster, a quantization cell, or a locality-sensitive hashing bucket. Let $c(z)$ denote the region assignment for tuple z. Within each region C_k, the sampling rate p_k can be chosen to balance storage and accuracy [19]. If sampling is Bernoulli with rate p_k within region k, then $\pi(z) = p_{c(z)}$. The challenge is to choose C_k and p_k so that the sample supports a broad class of queries while controlling worst-case error for tail segments.

Clustering-based regions can be derived from an offline embedding index on dimension entities such as products and sellers. This is attractive because the number of entities is smaller than the number of events, and because entity embeddings are reused across many fact tuples via joins. For example, a product embedding can be clustered into K groups, and then each event tuple inherits the cluster label of its product. This reduces the need to cluster event embeddings directly and yields strata aligned with stable entities [20]. The approach also supports schema evolution: even if event schemas change, the product key remains, and the embedding-based cluster can still be assigned.

However, clustering alone may not preserve rare outcome regions that matter for BI, such as high revenue orders, refunds, chargebacks, or policy violations. These outcomes may not correspond to distinct embedding clusters; they may be sparse within dense regions. To handle this, semantic sampling can incorporate outcome-aware oversampling. Let $u(z)$ denote an auxiliary score correlated with the magnitude or importance of $g(z)$, such as predicted revenue, predicted fraud risk, or a proxy like price [21]. The sampling rate can depend on both region and score, for instance via a piecewise schedule that increases p_k for high-score tuples. While this introduces

Configuration	Q-Error (p50)	Q-Error (p95)	Description
No semantics, random	1.38	2.94	Uniform sampling baseline
Schema-only	1.22	2.41	Partition-aware strata
Vector-only	1.15	2.03	Clustered on embeddings
Semantic + vector	1.08	1.74	Proposed hybrid method
Semantic + vector + reweigh	1.05	1.61	Calibration via importance weights

Table 5: Ablation study isolating the impact of semantic sampling and vector-similarity guided stratification.

Sample Fraction	Rel. Error @p95 (%)	Latency (s)	Observation
0.05%	11.9	1.9	Too noisy for tail KPIs
0.10%	7.6	2.6	Acceptable for exploratory dashboards
0.25%	4.1	3.4	Balanced trade-off for daily BI
0.50%	3.0	4.2	Near-exact for top-line metrics
1.00%	2.1	6.0	Marginal benefit, higher cost
2.00%	1.8	8.7	Close to exact scan behavior

Table 6: Effect of sample size on accuracy and latency for the hybrid semantic AQP engine.

unequal probabilities within regions, unbiased estimation remains possible.

A practical design in data lakes is a two-tier semantic sampling scheme. The first tier maintains a base sample per partition and region at a modest rate, ensuring broad coverage. The second tier maintains a targeted sample for rare or high-impact events, selected via a higher rate for tuples exceeding a score threshold or belonging to a monitored cohort. The union of the tiers forms the sample store, and the inclusion probability is the complement of the probability of being excluded from both tiers [22]. If tier decisions are independent conditional on tuple features, $\pi(z)$ can be computed analytically; if they are implemented as deterministic hash-based selections, $\pi(z)$ can be encoded as metadata.

To formalize, suppose tuple z belongs to region k and has score bin b. Define two inclusion decisions: a base decision with probability $p_k^{(0)}$ and a boost decision with probability $p_{k,b}^{(1)}$. If decisions are independent, the overall inclusion probability is

$$\pi(z) \;=\; 1 - \big(1 - p_k^{(0)}\big)\big(1 - p_{k,b}^{(1)}\big). \tag{2}$$

When implementing with deterministic hashing, $p_k^{(0)}$ and $p_{k,b}^{(1)}$ correspond to thresholds on independent hash functions. The sample store retains $\pi(z)$ or enough information to reconstruct it.

Semantic sampling must handle time [23]. Marketplace BI queries are heavily time-windowed, and distributional properties drift over time. If the sample is maintained per time partition, each partition can have its own semantic regions and rates, but this complicates query-time combination across partitions. Alternatively, regions can be defined globally over dimension entities, while rates can be partition-specific to reflect varying event volumes. A hybrid approach clusters dimension embeddings globally, assigns region labels consistently, and then chooses per-partition rates $p_{k,t}$ to satisfy storage budgets and accuracy targets. This retains semantic consistency while accommodating temporal variation.

Incremental maintenance is essential [24]. New products and sellers arrive and must be assigned to regions. This can be done by nearest centroid assignment in embedding space, which is fast and stable. If embeddings drift due to model retraining, centroids may shift. The framework treats embedding versions as part of metadata and supports dual labeling during transitions: tuples may carry both old and new region labels, enabling gradual migration. A more lightweight strategy uses locality-sensitive hashing, where region assignments are based on random projections and are relatively stable under small drift. Hash-based regions also simplify distributed computation because they avoid global clustering updates [25].

An additional constraint is join propagation. If events in the fact table reference products, and sampling is performed on events, the event inclusion probability may need to depend on the product region. This can be implemented by computing the product region label in the dimension table and joining it into the fact table as a derived column, either physically materialized or computed during ingestion. Many lakehouse pipelines already denormalize such derived attributes for performance. The semantic sampling layer thus becomes part of the ingestion pipeline, adding region labels and sampling flags [26].

While semantic sampling improves coverage for semantically defined predicates, it does not by itself en-

Index Type	Key Parameters	Build Time (min)	Peak Memory (GB)
IVF-Flat	8k lists, nprobe=16	37	12.4
IVF-PQ	8k lists, M=16, 8 bits	29	7.9
HNSW	$M = 32$, efConstruction=200	41	10.7
DiskANN	Beam=4, L=128	53	5.1
Hybrid IVF+HNSW	4k lists, $M = 24$	46	9.3

Table 7: Vector index configurations for similarity-guided stratification over semantic embeddings.

Query Class	Share of Workload	Typical Filters	Example Metric
Traffic overview	23%	Country, device, channel	Sessions per day
Funnel analysis	19%	Step, campaign, platform	Checkout completion rate
Merchandising	17%	Category, brand, price band	Gross merchandise value
Search quality	21%	Query cluster, rank position	Click-through rate
Experiment readouts	20%	Treatment, cohort, horizon	Lift in conversion rate

Table 8: Business intelligence workload mix used to evaluate the proposed approximate query engine.

sure low variance for a particular query. Queries vary widely, and a sampling design optimized for one region may waste effort for another. Therefore, the framework complements offline semantic sampling with query-time vector-similarity guided stratification, which selects and weights strata based on the query vector. The next section develops this query-time stratification and shows how it can be layered on top of maintained semantic samples.

4. Vector-Similarity Guided Stratification and Adaptive Allocation

Stratification reduces variance by partitioning the population into strata that are internally homogeneous with respect to the measure of interest and then sampling within each stratum [27]. In the marketplace setting, homogeneity is often semantic: products with similar embeddings may have similar conversion dynamics, price distributions, or fulfillment constraints. Yet a static stratification that is optimal for one measure may not be optimal for another, and static strata may not align with arbitrary semantic predicates. Vector-similarity guided stratification addresses this by using embedding similarity between the query representation and stratum representations to guide sampling allocation.

Assume the lake maintains a semantic partition $C_1, \ldots, C_K$ over tuples or over dimension entities propagated to tuples. For each stratum k, define a centroid vector $c_k \in \mathbb{R}^d$ and a weight W_k equal to the number of tuples in the stratum within the relevant time window. At query time, a query predicate P is mapped to a query vector $q = \psi(P)$ [28]. The similarity between q and stratum k is computed via a function $s(q, c_k)$, such as cosine similarity. This similarity is used in two related ways. It can define a soft relevance score indicating how much the stratum is expected to contribute to the query result, and it can approximate the conditional variance within the stratum by indicating how selective the predicate is within that stratum.

A typical semantic predicate is a similarity filter selecting tuples whose embedding is near a query vector, such as selecting products similar to a seed. In that case, strata near q will have higher selectivity. Even when the predicate is not a direct similarity filter, embeddings can still correlate with predicate satisfaction because embeddings encode attributes relevant to the predicate [29]. Therefore, similarity provides a computationally cheap proxy for selectivity and contribution.

To make this precise, let $I(z) = \mathbf{1}\{P(z)\}$ denote the predicate indicator. The stratum contribution to the sum aggregate is $A_k = \sum_{z \in C_k} g(z) I(z)$. The total is $A = \sum_k A_k$. If the sample store contains tuples from each stratum, the AQP engine can estimate A_k from the sample and sum them. The query-time question is how many sampled tuples to draw or scan from each stratum, given a fixed budget n of sample tuples to process. If the sample store is itself a fixed subset, the engine can choose which strata to read more fully by selecting additional sample partitions or higher-rate tiers, or by drawing additional tuples from a reservoir [30]. In many lakehouse contexts, the sample store is partitioned by stratum and time, enabling partial reads.

Classical optimal allocation for stratified sampling under a fixed total sample size allocates sample counts proportional to stratum weights and standard deviations. If within stratum k the standard deviation of $g(z)I(z)$ is σ_k, the Neyman allocation sets $n_k \propto W_k \sigma_k$. In practice, σ_k is unknown and varies by query. Vector-similarity guided stratification replaces σ_k with a query-dependent proxy $\hat{\sigma}_k(q)$ derived from similarity and from cached stra-

Metric	Baseline System	Proposed AQP Engine	Change
Median dashboard latency (s)	18.4	3.2	5.7× faster
95th percentile latency (s)	41.8	7.1	5.9× faster
Daily ad-hoc queries served	3.1k	9.6k	3.1× more
Data scanned per query (TB)	2.7	0.21	12.9× reduction
Compute cost per 1k queries ($)	112	37	67% lower
BI analyst satisfaction score	3.4/5	4.6/5	+1.2 points

Table 9: End-to-end business impact of deploying semantic sampling based approximate query processing.

tum statistics. A simple proxy is $\hat{\sigma}_k(q) = \alpha_k \cdot r_k(q)$, where α_k is a cached dispersion measure for g in stratum k and $r_k(q)$ is a relevance score computed from similarity.

One can define $r_k(q)$ as a squashed similarity: [31]

$$r_k(q) \;=\; \frac{\exp\big(\tau \cdot s(q, c_k)\big)}{\sum_{j=1}^{K} \exp\big(\tau \cdot s(q, c_j)\big)}, \tag{3}$$

where τ controls sharpness. This yields a distribution over strata, focusing allocation on those most similar to the query. Alternatively, $r_k(q)$ can be calibrated using historical queries by learning a mapping from similarity to observed selectivity. If the system logs predicate satisfaction rates per stratum for past queries, it can fit a monotone function that predicts $\Pr[I = 1 \mid k, q]$ from $s(q, c_k)$. The allocation can then reflect expected contribution $W_k \mathbb{E}[gI \mid k, q]$ and expected variance.

A more explicit model treats predicate satisfaction as a function of distance in embedding space. Suppose P includes a condition $d(\phi(z), q) \le \epsilon$, where d is a distance metric [32]. Within stratum k, the distribution of $\phi(z)$ can be approximated as a Gaussian around c_k with covariance Σ_k. Then the selectivity is approximately the probability mass of a ball around q, which depends on the Mahalanobis distance $(q - c_k)^\top \Sigma_k^{-1} (q - c_k)$. While exact computation is complex, the similarity $s(q, c_k)$ serves as a surrogate. The framework does not require Gaussian assumptions; it leverages similarity as a low-cost signal and relies on online feedback to correct misallocations.

Given a budget n, the engine selects n_k sample tuples from each stratum, where n_k is computed as

$$n_k \;=\; n \cdot \frac{W_k \hat{\sigma}_k(q)}{\sum_{j=1}^{K} W_j \hat{\sigma}_j(q)} \quad \text{with} \quad \hat{\sigma}_k(q) \;=\; \alpha_k \cdot r_k(q) + \beta_k \tag{4}$$

The term β_k prevents starvation, ensuring each stratum receives some coverage for unbiasedness and for detecting unexpected contributions [33]. In systems with a fixed precomputed sample rather than on-demand selection, n_k can be interpreted as the number of sampled tuples to scan from stratum k, possibly by scanning additional blocks until n_k tuples are processed.

Vector-guided stratification also helps with group-by. For a group key y, the distribution of y across strata can be skewed. If the query requests aggregates for many groups, the system must avoid missing groups that are concentrated in semantically distant strata. One strategy is to allocate a base budget across all strata for broad coverage and then allocate additional budget to strata similar to q to improve precision for the most relevant groups [34]. Another strategy uses multiple query vectors when the predicate includes multiple semantic components, such as a set of seed products. In that case, $r_k(q)$ can be defined as the maximum or average similarity over the set, or as a mixture distribution.

A further complication is that many BI predicates include structured filters that are not well represented by embeddings, such as exact seller identifiers or region codes. The framework treats such filters as restricting the set of eligible strata or as modifying stratum weights W_k by applying the structured filter on stratum metadata. For example, if strata are defined by product embeddings but the query filters to a seller cohort, the engine can estimate the fraction of each stratum belonging to the seller cohort using cached counts and adjust W_k accordingly [35]. This is feasible because structured filters often correspond to columns for which per-stratum histograms can be maintained.

The overall effect is that stratification becomes query-adaptive without requiring per-query precomputation. The engine uses a compact representation of strata, including centroids and summary statistics, and computes allocations from query vectors in milliseconds. The allocation then guides which portions of the semantic sample store to scan. The next section develops the estimation methods that operate on the resulting stratified sample, focusing on unbiasedness, variance estimation, and join-aware aggregates.

5. Estimators and Error Quantification for Semantic Stratified AQP

An AQP system is judged not only by point estimates but also by the credibility of its uncertainty quantification [36]. In marketplace BI, decisions often depend on detecting changes over time or differences between segments, and analysts need to know whether an apparent change is within noise. Therefore, the framework emphasizes estimators whose properties can be reasoned about under the sampling design and that admit practical variance estimation even when stratification is guided by embeddings.

Consider estimating a sum aggregate $A = \sum_{z \in \mathcal{U}} y(z)$, where $y(z) = g(z)\mathbf{1}\{P(z)\}$ and $\mathcal{U}$ is the population of joined tuples in the time window. Suppose the population is partitioned into K strata C_k with sizes N_k. Within stratum k, a sample S_k of size n_k is obtained, either by simple random sampling without replacement from the stratum population or by scanning a precomputed Bernoulli sample with inclusion probabilities. For clarity, first consider simple random sampling without replacement within each stratum [37]. The unbiased estimator of the stratum sum is

$$\hat{A}_k = N_k \cdot \bar{y}_k \quad \text{where} \quad \bar{y}_k = \frac{1}{n_k} \sum_{z \in S_k} y(z), \tag{5}$$

and the total estimator is $\hat{A} = \sum_k \hat{A}_k$. Under this design, the variance of $\hat{A}$ is

$$\mathrm{Var}(\hat{A}) = \sum_{k=1}^{K} N_k^2 \Big(1 - \frac{n_k}{N_k}\Big) \frac{S_k^2}{n_k}, \tag{6}$$

where S_k^2 is the finite-population variance of y in stratum k. This expression motivates allocating n_k proportional to $N_k S_k$. Vector-similarity guided allocation approximates this by using proxies for S_k that depend on the query vector.

In data lakes, sampling is often Bernoulli with inclusion probabilities $\pi(z)$, potentially varying across strata and tiers [38]. In that case, an unbiased estimator for the sum is the Horvitz–Thompson estimator

$$\hat{A} = \sum_{z \in S} \frac{y(z)}{\pi(z)}, \tag{7}$$

where S is the set of sampled tuples that satisfy the query-time scan conditions. If the sample is stratified with constant π_k within stratum k, then $\hat{A}_k = \sum_{z \in S_k} y(z)/\pi_k$. This aligns with scaling the sample mean by N_k when $\pi_k = n_k/N_k$. Unequal probabilities arise under outcome-aware oversampling; in that case, $\pi(z)$ varies within strata and must be accounted for.

Variance estimation under Bernoulli sampling can be obtained using standard unbiased or conservative estimators. For independent Bernoulli draws, a common variance estimator is [39]

$$\widehat{\mathrm{Var}}(\hat{A}) = \sum_{z \in S} \frac{1 - \pi(z)}{\pi(z)^2} y(z)^2, \tag{8}$$

which is unbiased when draws are independent. In partitioned data lakes, independence is often approximated by hash-based decisions. When sampling uses reservoir methods or sampling without replacement within blocks, the variance formula changes, but block-level approximations can be used by treating blocks as primary sampling units and applying cluster-robust estimators.

For ratio metrics such as averages and rates, the estimator must handle the ratio of two sums. Let $T = \sum_z y(z)$ and $B = \sum_z b(z)$, where $b(z) = \mathbf{1}\{P(z)\}$ for an average over qualifying tuples or $b(z) = w(z)\mathbf{1}\{P(z)\}$ for weighted averages. The metric is $\theta = T/B$ [40]. Using Horvitz–Thompson estimators $\hat{T}$ and $\hat{B}$, the plug-in estimator $\hat{\theta} = \hat{T}/\hat{B}$ is approximately unbiased for large samples. Its variance can be approximated via a delta method:

$$\mathrm{Var}(\hat{\theta}) \approx \frac{1}{B^2}\mathrm{Var}(\hat{T}) + \frac{T^2}{B^4}\mathrm{Var}(\hat{B}) - \frac{2T}{B^3}\mathrm{Cov}(\hat{T}, \hat{B}). \tag{9}$$

In stratified designs where T and B are estimated from the same sampled tuples, the covariance term can substantially reduce variance. The system can estimate these components online using per-tuple contributions. Under independent Bernoulli inclusion,

$$\widehat{\mathrm{Cov}}(\hat{T}, \hat{B}) = \sum_{z \in S} \frac{1 - \pi(z)}{\pi(z)^2} y(z)\, b(z). \tag{10}$$

These expressions provide error bars that can be displayed in BI dashboards, and they support comparisons between segments by propagating uncertainty [41].

Group-by aggregates introduce multiple estimands θ_y. A naive approach runs the above estimators per group, but small groups can have few sampled tuples, yielding unstable estimates and misleading variance estimates. Semantic stratification helps because groups often align with semantic clusters, but not always. The framework adopts two stabilizing mechanisms. The first is a minimum effective sample size rule based on estimated $\hat{B}_y$: if $\hat{B}_y$ is

too small, the system either increases sampling budget for strata that contain group y or marks the estimate as low-confidence. The second is shrinkage for variance estimation, where per-group variance estimates are smoothed toward stratum-level variance estimates to reduce noise in tails. This smoothing is performed in the uncertainty layer rather than altering point estimates, maintaining unbiasedness while providing more stable intervals [42].

Join-aware estimation is critical. Many queries join a large event fact table to a product dimension to filter by product attributes or embeddings. If sampling occurs on the fact table, the inclusion probability of an event may depend on the product region. The Horvitz–Thompson estimator remains valid as long as $\pi(z)$ reflects the inclusion probability of the joined tuple. When the join is one-to-many or many-to-many, such as joining sessions to multiple attributed intents, the population definition must be clarified to avoid double counting [43]. In BI, these joins are often defined with specific semantics, such as attributing an order to the last click. The sampling design must follow the same semantics. If the join duplicates tuples, the measure function g implicitly includes the attribution weight, and the estimator applies to the expanded population.

A related challenge arises when the semantic predicate is evaluated using embeddings stored in a feature store. If embeddings are missing for some tuples, the predicate may treat them as non-qualifying or may impute defaults [44]. Missingness can bias estimates if correlated with the measure. The framework treats missing embeddings as a separate stratum C_0 with its own sampling rate and statistics. This ensures that missingness is represented in the sample and that its contribution is explicitly estimated rather than silently excluded.

Vector-similarity guided allocation influences n_k and thus the variance, but it must not bias estimates. Bias can occur if the allocation changes inclusion probabilities in a way that is not accounted for [45]. The framework avoids this by separating selection of which stored sample tuples to scan from the sampling design that determines inclusion probabilities. If the sample store is precomputed with known $\pi(z)$, then scanning a subset of the sample store creates an additional stage of sampling. To preserve unbiasedness, the second-stage selection must be incorporated. One approach is to treat scanned tuples as a subsample of the stored sample with known scan probability ρ_k per stratum, yielding overall inclusion probability $\pi(z)\rho_{c(z)}$. The estimator then uses weights $1/(\pi(z)\rho_{c(z)})$. In practice, ρ_k can be controlled by scanning a random subset of sample files per stratum using hash selection. This enables query-time budget allocation without sacrificing unbiasedness.

When query-time selection is deterministic, such as scanning the first n_k tuples in a stratum, unbiasedness can fail due to ordering effects [46]. Therefore, the system stores sample tuples in randomized order within stratum partitions or uses random access selection via hashing. Data lake storage formats can support this by writing sample data with shuffled ordering during compaction jobs.

Finally, error quantification must handle distribution shift. Embeddings drift, product catalogs change, and seasonal effects alter behavior. Variance estimates based on within-stratum sample variance remain valid for the current query execution but may be misleading if the sample store lags behind the population due to delayed ingestion [47]. The framework includes freshness metadata per stratum and time partition and inflates uncertainty when freshness is low. It also monitors discrepancies between sample-derived selectivity predictions and observed selectivity in scanned tuples, using these discrepancies to adjust allocation and to flag queries where semantic similarity is a poor proxy. With these estimation foundations, the next section turns to system design and practical integration in a lakehouse environment.

6. System Architecture, Integration, and Operational Considerations

Deploying semantic AQP in a marketplace data lake requires integration with ingestion pipelines, catalogs, feature stores, and query engines. The architecture can be conceptualized as three layers: a semantic metadata layer that provides embeddings and region assignments, a sample storage layer that maintains semantic samples aligned with lake partitions, and a query-time AQP planner that performs vector-guided allocation and estimation [48].

The semantic metadata layer exposes embedding functions and versions. For entities such as products, sellers, and queries, embeddings are computed by upstream ML pipelines and stored in a feature store keyed by entity identifiers and version. The AQP system requires access to embeddings for assigning region labels and for computing stratum centroids. For fact tuples, embeddings are typically inherited via joins to entities, avoiding per-event embedding computation. The metadata layer also stores region definitions, such as cluster centroids or hash projection parameters, and maps each entity to a region label. This mapping can be materialized as an additional column in dimension tables and propagated to fact tables during ingestion or via late materialization at query time

[49]. Materialization is often preferable for performance because it enables partition pruning and reduces join cost for region assignment.

The sample storage layer maintains sample tables per fact relation and time partition. These sample tables contain a subset of tuples along with sampling metadata including stratum label, inclusion probability components, and any required join keys. To support query-time selective scanning by strata, the sample tables are physically partitioned or clustered by stratum label. In a lakehouse, this can be implemented by writing separate files per stratum within each time partition, or by using clustering features that colocate similar stratum labels [50]. Compaction jobs periodically rebalance file sizes and randomize tuple ordering within files to support unbiased subsampling during query-time scans.

Maintaining semantic samples incrementally involves decisions about when to compute region labels and when to apply sampling. A common ingestion pattern is append-only writes to raw bronze tables followed by transformations into curated silver and gold tables. The sampling layer can operate at the silver stage, where schemas are stabilized and join keys are available. During this stage, tuples receive region labels by joining to dimension tables that contain region mappings, and sampling decisions are applied using deterministic hashes [51]. The resulting sampled tuples are written to sample tables. Because sampling decisions are deterministic, late-arriving data can be sampled consistently without changing prior inclusion probabilities.

Operationally, the system must allocate a storage budget for sample tables. Semantic sampling with multiple tiers increases storage, but the budget can be controlled by choosing base rates and boost rates. The system monitors per-stratum volumes and can adjust rates over time to maintain budget, treating rate changes as a versioned policy [52]. When rates change, inclusion probabilities for new data change, but estimators remain valid as long as $\pi(z)$ is stored per tuple or reconstructible from policy versions. For queries spanning long time windows, the estimator naturally combines tuples with different π values.

The query-time AQP planner integrates with the SQL engine or BI layer. When a query is submitted, the planner determines whether it is eligible for AQP, based on query type, required accuracy, and governance policies. For eligible queries, the planner rewrites the query to operate on sample tables and to apply appropriate weighting. For example, a sum aggregate becomes a weighted sum over sampled tuples, and filters and joins are applied as usual [53]. If the query includes semantic similarity predicates, the planner computes the query vector q and uses it to compute relevance scores $r_k(q)$ over strata. It then chooses scan probabilities ρ_k or scan budgets n_k per stratum, subject to a latency or cost budget. This information is passed to the execution engine as constraints, such as selecting a random subset of files per stratum or limiting scanned rows via sampling functions.

A critical concern is that BI workloads often involve many concurrent queries, and the AQP system must avoid creating unpredictable load. Vector-guided allocation can concentrate scanning on a few strata, which is efficient for single queries but can create hotspots if many queries target similar regions, such as during a trending event [54]. The architecture therefore includes a resource governor that caps per-stratum concurrent scan throughput and falls back to broader scanning when hotspots occur. This can slightly increase variance but improves system stability. The governor can also cache intermediate results for repeated queries, such as stratum-level summaries that can be reused across dashboards.

Embedding drift and region maintenance are ongoing operations. When embeddings are retrained, region assignments can change [55]. The system treats region definitions as versioned artifacts and can support coexisting versions. During a transition period, new tuples are labeled with the new version while old tuples retain old labels. The AQP planner can either operate within a single version for a given query window or map between versions by maintaining a correspondence between old and new regions, such as via nearest centroid mapping. A conservative approach is to base stratification on stable hash-based regions that are less sensitive to drift, while using centroids for similarity scoring. This reduces maintenance cost at the expense of slightly less coherent strata.

Another operational issue is semantic coverage for new entities [56]. Cold-start products and sellers may have embeddings that are low quality or missing. The system isolates these cases into explicit strata and may oversample them to ensure visibility. For BI, cold-start segments can be important, such as monitoring onboarding success for new sellers. Semantic sampling thus aligns with operational needs beyond AQP.

Governance and privacy constraints must be respected [57]. Sample tables are derived data and must inherit access controls. If the lake enforces row-level or column-level security, the sample tables must be secured similarly. Weighted estimation must not leak sensitive information through small strata. The system can enforce minimum aggregation thresholds, suppressing or coarsening results for groups with low effective population sizes. Because AQP

can increase the risk of disclosure if it enables inference from small samples, the system may require differential privacy mechanisms in certain contexts [58]. The framework described here does not assume differential privacy, but it is compatible with adding noise calibrated to sensitivity after computing weighted aggregates, provided that the noise is accounted for in uncertainty reporting.

Finally, the system must interact with exact computation. Some queries require exact answers, and some approximate results may be flagged as low confidence. The architecture supports hybrid execution where the AQP engine produces an approximate answer quickly and then optionally triggers an exact computation path if required by policy. Within this paper, the focus remains on the AQP path, but in practice, coexistence with exact pipelines improves analyst trust by enabling spot checks and by providing ground truth for calibration of similarity-based allocation.

With architecture in place, the effectiveness of the approach depends on empirical trade-offs: how much variance reduction semantic stratification yields, how robust similarity proxies are across query types, and how maintenance cost scales [59]. The next section synthesizes these considerations in an evaluation-oriented discussion, focusing on marketplace-relevant scenarios and on the mechanisms by which semantic sampling and vector-guided allocation improve accuracy per unit cost.

7. Evaluation Considerations and Marketplace-Oriented Analysis

Assessing semantic AQP for marketplace BI requires evaluating both statistical efficiency and operational efficiency. Statistical efficiency concerns how much error decreases for a fixed scanned sample size, especially for semantically filtered queries and long-tail segments. Operational efficiency concerns ingestion overhead, sample storage cost, query planning overhead, and runtime predictability. While specific numeric results depend on datasets and implementation details, the evaluation can be structured around representative workloads and measurable outcomes [60].

A representative workload includes dashboard-style aggregates such as gross merchandise value, units, conversion rate, and refund rate, sliced by dimensions like category, brand, seller tier, region, and fulfillment method. It also includes exploratory queries such as identifying categories with unusual conversion drop for products similar to a seed item, measuring the performance of semantically defined cohorts such as eco-friendly products inferred from text, and monitoring risk metrics for clusters of listings similar to known policy violations. These queries stress semantic predicates and require reliable estimation in the tail.

The baseline comparisons typically include uniform sampling of the fact table, stratified sampling by common structured keys such as category or region, and potentially stratified sampling by hash buckets unrelated to semantics. Uniform sampling tends to provide stable estimates for head segments but can have high relative error for tail segments and for predicates that select semantically coherent but small subsets [61]. Key-based stratification can improve accuracy when the key aligns with the predicate, but it often fails when semantic concepts cut across keys or when keys are noisy due to catalog inconsistencies. Semantic sampling aims to improve coverage for semantic predicates by aligning strata with embedding neighborhoods that capture latent attributes.

Variance reduction can be analyzed through the lens of within-stratum homogeneity. If the measure $y(z)$ varies smoothly in embedding space, then clustering by embeddings reduces within-stratum variance S_k^2 compared to arbitrary partitions. For a fixed sample budget, this reduces $\mathrm{Var}(\hat{A})$. The extent of improvement depends on how well embeddings capture the factors that drive y [62]. In marketplaces, embeddings trained for retrieval and recommendation often capture product similarity and intent similarity, which correlate with conversion patterns and price distributions, though not perfectly. Therefore, improvements are expected to be strongest for measures that are semantically driven, such as conversion rate, add-to-cart rate, or relevance-driven engagement, and weaker for measures dominated by operational constraints unrelated to semantics, such as carrier delays driven by weather.

Vector-similarity guided allocation further improves efficiency by concentrating sampling on relevant strata for a given query. The mechanism is especially beneficial when the predicate induces strong selectivity that varies across strata. For example, a query selecting products similar to a seed will concentrate qualifying tuples in strata near the seed in embedding space. Uniform allocation across strata wastes budget on irrelevant strata with near-zero selectivity, increasing variance because the effective sample size for qualifying tuples is small [63]. Similarity-guided allocation increases the effective sample size by scanning more from relevant strata, thus reducing variance of T and B in ratio metrics. The benefit depends on the calibration of similarity to selectivity. If similarity is poorly correlated with selectivity for a predicate, allocation may misfire. The framework mitigates this by maintaining a

nonzero base allocation and by using online feedback to adjust.

Tail behavior is central [64]. Consider estimating refund rate for a niche cluster of products, where the denominator is small and the numerator even smaller. Uniform sampling may yield zero observed refunds, producing a point estimate of zero with misleading confidence if variance estimation is naive. Semantic oversampling of high-risk or high-value outcomes increases the chance of observing relevant events, while maintaining unbiasedness through weighting. Embedding-aware strata also help because niche clusters correspond to localized regions, making it feasible to allocate more sampling to those regions when queries target them. In contrast, key-based stratification may not isolate the niche cluster if categories are broad [65].

Group-by queries can be evaluated by measuring error distribution across groups. A key metric is the fraction of groups whose relative error falls below a threshold. Semantic sampling is expected to reduce the number of groups with extreme errors by improving representation of rare groups that occupy distinct semantic regions. Vector-guided allocation can further improve accuracy for groups relevant to a query, but it can also reduce accuracy for unrelated groups if budget is too concentrated. This trade-off is acceptable in interactive exploration when the analyst focuses on a semantic cohort, but it is less acceptable for global dashboards. Therefore, evaluation should separate cohort-focused queries from global queries and measure the impact of allocation policies [66]. A balanced policy maintains global coverage while enabling targeted refinement.

Join effects can be evaluated by comparing queries executed on denormalized tables versus join-based execution. If semantic region labels are propagated correctly, sampling on the fact table should match sampling on the joined population. Discrepancies arise when dimension attributes are missing or when join keys are inconsistent. Semantic sampling can sometimes mask data quality issues by smoothing across embeddings, which is useful for BI robustness but can also hide anomalies [67]. Evaluation should therefore include data quality stress tests, such as missing embeddings, incorrect category labels, and rapidly changing catalogs.

Operationally, ingestion overhead is measured by the added cost of computing region labels and sampling decisions. If region labels are computed via joins to a dimension mapping table, the overhead is proportional to join cost, which is often already incurred for enrichment. Sampling decisions via hashing are negligible. The main overhead is writing sample tables and maintaining their partitioning [68]. Compaction and clustering jobs add cost but can be scheduled during low-traffic periods. Storage overhead is measured as the ratio of sample table size to base table size, controlled by sampling rates. For many BI workloads, a sample size of a few percent can yield substantial speedups, but the optimal rate depends on required accuracy and on query diversity.

Query planning overhead includes computing query vectors and relevance scores across K strata. If K is large, naive computation is expensive [69]. The system can accelerate by indexing centroids in an approximate nearest neighbor structure and computing relevance over only the top similar strata, with a background mass assigned to the rest via the base allocation. This reduces planning latency and avoids scanning many irrelevant strata. The approximation affects allocation but not unbiasedness if scan probabilities are accounted for; it affects variance by potentially missing moderately relevant strata. Evaluation should measure sensitivity to the number of considered strata.

Predictability is crucial for BI. AQP should reduce latency without causing large variance in runtime [70]. Vector-guided allocation can create runtime variability if it chooses to scan many strata for some queries and few for others. The resource governor and budget constraints mitigate this by bounding scan volume. Evaluation should measure tail latencies and how often the system falls back to broader scanning due to hotspots. It should also measure stability of approximate answers across repeated executions, which can be improved by using deterministic subsampling within strata, though that must be balanced against unbiasedness and against the need for independent samples when estimating uncertainty.

A final evaluation axis is robustness under drift [71]. Marketplace behavior shifts with seasonality, promotions, and external events. Embeddings retrain and change. Semantic stratification must remain useful under these shifts. Evaluation can simulate drift by training embeddings on one period and applying them to later periods, measuring how variance reduction degrades. The dual-version labeling strategy can be evaluated by comparing transition smoothness, examining whether estimates remain consistent across version boundaries [72]. The system's online calibration can be evaluated by measuring how quickly it adapts allocation when similarity-selectivity relationships change.

These evaluation considerations collectively indicate where semantic sampling and vector-similarity guided stratification are most valuable. They are particularly

suited to workloads where semantic predicates are common, where tail segments matter, and where embeddings are already part of the marketplace stack. They may be less beneficial for purely structured queries over stable low-cardinality dimensions, where classical stratification suffices, or when embeddings are low quality for the domain in question. The concluding section summarizes implications, limitations, and directions for refinement within the constraints of data lake operations.

8. Conclusion

Approximate query processing in marketplace data lakes must contend with semantic heterogeneity, high cardinality, and long-tail decision relevance [73]. AQP designs that treat the data lake as a purely structured relational store often struggle because the subpopulations that matter are frequently defined by latent semantic concepts expressed through text, images, and behavioral signals, and because these concepts evolve over time. The framework developed in this paper addresses these realities by combining semantic sampling with vector-similarity guided stratification. Semantic sampling maintains embedding-aware samples that preserve representation across semantic neighborhoods and rare but impactful outcome regions, enabling approximate answers for a broad class of BI aggregates without requiring enumerating all possible segment combinations. Vector-similarity guided stratification complements this by using query representations in embedding space to allocate query-time sampling effort toward strata most relevant to the predicate, improving effective sample size and reducing variance for semantically filtered queries.

The technical development highlights how unbiased estimation can be retained under embedding-aware designs by tracking inclusion probabilities and, when query-time scanning selects subsets of stored samples, incorporating second-stage scan probabilities [74]. Stratified and Horvitz–Thompson estimators provide a foundation for sums and ratios, while online variance estimation supports uncertainty reporting in dashboards. The framework also emphasizes join-aware considerations, recognizing that many semantic attributes live on dimension entities and must be propagated to facts for efficient sampling and stratification. Operational integration in lakehouse settings is feasible through partition-aligned sample tables, versioned region definitions, and incremental maintenance that leverages existing enrichment pipelines.

Limitations remain. Embedding similarity is an imperfect proxy for query selectivity and variance, and its usefulness varies across measures and domains [75]. Allocation policies must balance targeted refinement with broad coverage to avoid degrading accuracy for unrelated groups in global dashboards. Embedding drift and missingness require explicit handling to avoid silent bias. Moreover, governance constraints can restrict sampling and approximate reporting in sensitive contexts, motivating additional protections such as thresholding and, where required, privacy-preserving noise addition. Within these constraints, semantic AQP is best viewed as a pragmatic layer that complements exact computation, improving latency and interactivity for many BI workloads while providing mechanisms to detect when approximation is unreliable.

Future refinements can focus on learning better similarity-to-selectivity calibrations from query logs, incorporating multi-modal embeddings and uncertainty in embeddings themselves, and designing adaptive maintenance policies that allocate sampling storage to strata based on observed query demand and business impact. The broader implication is that as data lakes increasingly incorporate semantic representations as first-class metadata, AQP can exploit this structure to move beyond key-based sampling, yielding approximate analytics that better align with the semantic nature of marketplace decision making [76].

References

[1] X. Yuan and J. Yang, "Asplos - effective concurrency testing for distributed systems," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 1141–1156, ACM, 3 2020.

[2] J. Chen, C. Du, P. Han, and X. Du, "Real-time digital simulator for distributed systems:," *SIMULATION*, vol. 97, pp. 003754972098686–309, 1 2021.

[3] 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.

[4] "Universal constructions in algebraic specification of distributed systems," *Informatics and Applications*, 4 2025.

[5] *On Strict Homogeneous Lyapunov Function for Generalized Homogeneous PI Controller*, 9 2021.

[6] R. Malik, S. Kim, X. Jin, C. Ramachandran, J. Han, I. Gupta, and K. Nahrstedt, "Mlr-index: An in-

dex structure for fast and scalable similarity search in high dimensions," in *International Conference on Scientific and Statistical Database Management*, pp. 167–184, Springer, 2009.

[7] R. R, R. S. Gowda, P. N, S. Kiran, L. B. S, and C. K. H. A, "Training in a large language model for distributed systems," in *2025 5th International Conference on Pervasive Computing and Social Networking (ICPCSN)*, pp. 1294–1298, IEEE, 5 2025.

[8] A. Bretas and A. Dutta, "Cyber-physical power systems protection: The byzantine cybersecurity framework," in *2023 IEEE Kansas Power and Energy Conference (KPEC)*, pp. 1–5, IEEE, 4 2023.

[9] D. Schneider and B. Uekermann, "Efficient partition-of-unity radial-basis-function interpolation for coupled problems," *SIAM Journal on Scientific Computing*, vol. 47, pp. B558–B582, 4 2025.

[10] R. P. M, "Dynamic load balancing in distributed systems," *International Journal for Research in Applied Science and Engineering Technology*, vol. 13, pp. 694–701, 9 2025.

[11] S. Pallewatta, V. Kostakos, and R. Buyya, "Placement of microservices-based iot applications in fog computing: A taxonomy and future directions," *ACM Computing Surveys*, vol. 55, pp. 1–43, 7 2023.

[12] E. A. Saksonov, Y. L. Leokhin, and V. N. Azarov, "Information interaction in distributed systems," in *2020 International Conference Quality Management, Transport and Information Security, Information Technologies (IT&QM&IS)*, pp. 15–19, IEEE, 9 2020.

[13] K. M. Goeschka, R. P. S. de Oliveira, P. Pietzuch, and G. Russello, "Session details: Theme: Distributed systems: Dads - dependable, adaptive, and secure distributed systems track," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, ACM, 4 2019.

[14] N. V. Patel, "Applying synthetic control methods to address causal identification challenges in the ride-hailing industry," *Journal of Data Science, Predictive Analytics, and Big Data Applications*, vol. 8, no. 7, pp. 27–49, 2023.

[15] N. S. Chatharajupalli, R. Rotta, R. Karnapke, and J. Nolte, "Vibromote: Wi-fi-based mesh communication for railway bridge inspection and monitoring," in *2025 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–6, IEEE, 10 2025.

[16] A. Mishra, "Global convergence in asynchronous distributed systems," 2 2020.

[17] M. Reisecker, C. Marcelino, T. Pusztai, and S. Nastic, "Gaia: Hybrid hardware acceleration for serverless ai in the 3d compute continuum," in *Proceedings of the IEEE/ACM 12th International Conference on Big Data Computing, Applications and Technologies*, pp. 1–10, ACM, 12 2025.

[18] W. Wu, "Accelerating distributed systems by in-network computing," in *Proceedings of the ACM CoNEXT Workshop on In-Network Computing and AI for Distributed Systems*, pp. 1–2, ACM, 11 2025.

[19] Y. Zhou, W. Dai, Y. Zhang, K.-K. R. Choo, X. Xie, and D. Zou, "Efficient and privacy-preserving artificial neural network model training with separated data in iot," *IEEE Internet of Things Journal*, vol. 12, pp. 37050–37061, 9 2025.

[20] B. Sedlak, V. C. Pujol, P. K. Donta, and S. Dustdar, "Active inference on the edge: A design study," in *2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pp. 550–555, IEEE, 3 2024.

[21] G. A. M. Sborz, G. A. Pohl, F. Viel, and C. A. Zeferino, "Sbcci - a custom processor for an fpga-based platform for automatic license plate recognition," in *Proceedings of the 32nd Symposium on Integrated Circuits and Systems Design*, pp. 15–6, ACM, 8 2019.

[22] 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.

[23] D. G. Balreira, T. da Silva Araújo, and F. Petrillo, "Visualizing kubernetes distributed systems: An exploratory study," in *2023 IEEE Working Conference on Software Visualization (VISSOFT)*, pp. 12–22, IEEE, 10 2023.

[24] A. Samanta, D. Karnehm, A. Neve, and S. Williamson, "Data-informed healthy cell clustering technique for second-life applications

of retired electric vehicle batteries," in *2024 IEEE Transportation Electrification Conference and Expo (ITEC)*, pp. 1–5, IEEE, 6 2024.

[25] S. Gupta, R. Verma, and N. Dhanda, "Introduction to next-generation internet and distributed systems," 7 2024.

[26] "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.

[27] Z. Dong, Z. Wang, X. Zhang, X. Xu, C. Zhao, H. Chen, A. Panda, and J. Li, "Fine-grained re-execution for efficient batched commit of distributed transactions," *Proceedings of the VLDB Endowment*, vol. 16, pp. 1930–1943, 6 2023.

[28] K. Marszałek, A. Domański, R. Cupek, and M. Drewniak, "Testing quality of service of communication system for agv fleet with software-defined network," in *2023 IEEE International Conference on Big Data (BigData)*, pp. 5078–5086, IEEE, 12 2023.

[29] 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.

[30] N. V. Patel, "Estimating the causal effects of pricing interventions in two-sided marketplaces under interference and spillovers," *Frontiers in Health Informatics*, vol. 13, no. 8, pp. 7572–7601, 2024.

[31] "Demystifying distributed systems in cloud-native environments," *Journal of Computational Analysis and Applications*, vol. 34, 10 2025.

[32] E. Feng, D. Feng, D. Du, Y. Xia, and H. Chen, "snpu: Trusted execution environments on integrated npus," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 708–723, IEEE, 6 2024.

[33] M. Krzysztoń, B. Bok, P. Żakieta, and J. Kołodziej, "Evasion attacks on ml in domains with nonlinear constraints," in *2024 IEEE 24th International Symposium on Cluster, Cloud and Internet Computing Workshops (CCGridW)*, pp. 112–119, IEEE, 5 2024.

[34] A. Rogan, A. Kolar-PoŻun, 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.

[35] Y. Mnaoui, A. Najoua, and H. Ouajji, *Automatic English Proficiency Assessment for Air Traffic Controllers in Emergencies*, pp. 996–1002. Springer International Publishing, 2 2022.

[36] Z. Mayransaev and A. B. Chernyshev, "Generalized circular criterion for the absolutesustainability of distributed systems," *IZVESTIYA SFedU. ENGINEERING SCIENCES*, pp. 182–189, 7 2021.

[37] S. E, "Logical and vector clocks used in distributed systems," *Journal of emerging technologies and innovative research*, vol. 8, 5 2021.

[38] 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.

[39] 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.

[40] E. Tangmunchittham and K. Piromsopa, "An analysis of python serialization towards distributed systems," in *2022 19th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pp. 1–4, IEEE, 5 2022.

[41] G. Talmale and U. Shrawankar, "Cluster based real time scheduling for distributed system," *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 10, pp. 137–156, 3 2021.

[42] L. A. Vlasenko, A. G. Rutkas, and A. A. Chikrii, "Stochastic differential games in distributed systems with delay," *Journal of Automation and Information sciences*, vol. 1, pp. 41–54, 1 2021.

[43] 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.

[44] P. Pinter, A. Morichetta, and S. Dustdar, "Distributed model serving for real-time opinion detection," in *2024 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pp. 64–73, IEEE, 7 2024.

[45] V. Kale, *Distributed Systems*, pp. 159–182. CRC Press, 10 2018.

[46] S. L. S. Ortiz, J. G. R. Enriquez, W. M. Tan, and C. A. M. Festin, "Hotspotter: An incentivized crowdsensing system for wifi and cellular network coverage visualization," in *2024 IEEE 29th Asia Pacific Conference on Communications (APCC)*, pp. 542–548, IEEE, 11 2024.

[47] 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.

[48] A. Kolar-Požun, M. Jančič, M. Rot, and G. Kosec, *Oscillatory Behaviour of the RBF-FD Approximation Accuracy Under Increasing Stencil Size*, pp. 515–522. Germany: Springer Nature Switzerland, 6 2023.

[49] N. Ghasem, *Theory and Applications of Distributed Systems*, pp. 107–153. CRC Press, 11 2018.

[50] A. Hojjat, J. Haberer, T. Zainab, and O. Landsiedel, "Limitnet: Progressive, content-aware image offloading for extremely weak devices & networks," in *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*, pp. 519–533, ACM, 6 2024.

[51] Y. Tyryshkina, "Understanding join strategies in distributed systems," in *2021 International Seminar on Electron Devices Design and Production (SED)*, pp. 1–4, IEEE, 4 2021.

[52] J. Yuan, A. Le-Tuan, M. Nguyen-Duc, T.-K. Tran, M. Hauswirth, and D. Le-Phuoc, *VisionKG: Unleashing the Power of Visual Datasets via Knowledge Graph*, pp. 75–93. Germany: Springer Nature Switzerland, 5 2024.

[53] M. A. Helcig and S. Nastic, "Fedccl: Federated clustered continual learning framework for privacy-focused energy forecasting," in *2025 IEEE 9th International Conference on Fog and Edge Computing (ICFEC)*, pp. 50–57, IEEE, 5 2025.

[54] J. A. Hamilton, D. A. Nash, and U. W. Pooch, *Distributed Systems*, pp. 147–182. CRC Press, 8 2020.

[55] M. Singh, "Optimization methods for high-load distributed systems," *International Journal of Latest Engineering and Management Research (IJLEMR)*, vol. 10, pp. 32–36, 3 2025.

[56] 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.

[57] J. D. Herath, P. Yang, and G. Yan, "Codaspy - real-time evasion attacks against deep learning-based anomaly detection from distributed system logs," in *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, pp. 29–40, ACM, 4 2021.

[58] 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.

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

[60] S. Fan, Z. Hua, Y. Xia, and H. Chen, "Xputee: A high-performance and practical heterogeneous trusted execution environment for gpus," *ACM Transactions on Computer Systems*, vol. 43, pp. 1–27, 4 2025.

[61] C. Akmut, "6.824 'distributed systems' (special 4)," 12 2023.

[62] S. Periyasamy, P. Kaliyaperumal, A. Alagarsamy, T. Elumalai, and T. Karuppiah, "A framework for dynamic monitoring of distributed systems featuring adaptive security," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 37, pp. 660–660, 1 2025.

[63] L. F. H. Duarte, G. B. Nardes, W. Grignani, D. R. Melo, and C. A. Zeferino, "Deep nibble: A 4-bit number format for efficient dnn training and inference in fpga," in *2024 37th SBC/SBMicro/IEEE Symposium on Integrated Circuits and Systems Design (SBCCI)*, pp. 1–5, IEEE, 9 2024.

[64] "Distributed system based on correlation sequencing algorithm," *Distributed Processing System*, vol. 2, 9 2021.

[65] N. V. Patel, "Estimating heterogeneous treatment effects of driver incentives in ride-hailing platforms," *Northern Reviews on Algorithmic Research, Theoretical Computation, and Complexity*, vol. 9, no. 11, pp. 16–43, 2024.

[66] 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.

[67] S. Ghosh, *Primer on Web3 and Distributed Systems*, pp. 33–67. Productivity Press, 8 2024.

[68] U. A. Kuehn, "A decentralized iot payment service for digital currencies," in *2025 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pp. 1–4, IEEE, 8 2025.

[69] P. Chakraborty, *Distributed Systems*, pp. 437–557. Chapman and Hall/CRC, 10 2023.

[70] Y. Simmhan, "Pedagogic practices for teaching distributed systems courses," in *Proceedings of the 15th Annual ACM India Compute Conference*, pp. 6–6, ACM, 11 2022.

[71] F. Strnisa, M. Jancic, and G. Kosec, "A meshless solution of a small-strain plasticity problem," in *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*, pp. 257–262, IEEE, 5 2022.

[72] P. Raith, T. Rausch, A. Furutanpey, and S. Dustdar, "faas-sim: A trace-driven simulation framework for serverless edge computing platforms," *Software: Practice and Experience*, vol. 53, pp. 2327–2361, 10 2023.

[73] 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.

[74] "International journal of distributed systems and technologies," 8 2024.

[75] T. Pusztai, J. Hisberger, C. Marcelino, and S. Nastic, "Stardust: A scalable and extensible simulator for the 3d continuum," in *2025 IEEE International Conference on Edge Computing and Communications (EDGE)*, pp. 44–53, IEEE, 7 2025.

[76] M. H. Hilman, M. A. Rodriguez, and R. Buyya, "Multiple workflows scheduling in multi-tenant distributed systems: A taxonomy and future directions," 9 2018.