

StreamPLAID: Efficient Incremental Indexing via Monotonic Radius Hierarchical Partitioning for Multi-Vector Retrieval

Chunming MA
Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, Guangdong, China
cma859@connect.hkust-gz.edu.cn

Xujia Li*
The Hong Kong University of Science and Technology
Hong Kong SAR, China
leexujia@ust.hk

Quanqing Xu
OceanBase, Ant Group
Hangzhou, China
xuquanqing.xqq@oceanbase.com

Chuanhui Yang
OceanBase, Ant Group
Hangzhou, China
rizhao.ych@oceanbase.com

Lei Chen
Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, Guangdong, China
leichen@cse.ust.hk

ABSTRACT

Multi-Vector Retrieval (MVR) provides crucial infrastructural support for modern Large Language Model (LLM) applications, enabling state-of-the-art semantic matching for Retrieval-Augmented Generation (RAG). However, existing MVR indexes heavily rely on static partitioning frameworks. While effective for offline construction, they incur prohibitive maintenance costs in dynamic settings, leading to severe write amplification or degraded performance under continuous data distribution shifts. To address these bottlenecks, we propose StreamPLAID, an efficient incremental indexing framework designed for streaming MVR workloads. StreamPLAID introduces Monotonic Radius Hierarchical Partitioning (MRHP), transforming flat clustering into a multi-layer architecture with a high-probability $\mathcal{O}(1)$-approximation guarantee for centroid optimality. Building upon MRHP to handle continuous data streams, we propose two novel mechanisms: *Incremental Update by Streaming Routing* and *Topology-Aware Subset Extraction and Localized Repair*. These strategies enable the system to handle updates locally and seamlessly reconstruct degraded partitions without triggering expensive global re-clustering, theoretically reducing the per-update index maintenance complexity from $\mathcal{O}(N\sqrt{N})$ to $\mathcal{O}(1)$ with respect to the global corpus size under bounded localized repair. Extensive evaluations demonstrate that StreamPLAID achieves substantial improvements in index maintenance efficiency while preserving competitive retrieval accuracy under both in-domain and out-of-domain incremental datasets.

PVLDB Reference Format:
Chunming MA, Xujia Li, Quanqing Xu, Chuanhui Yang, and Lei Chen. StreamPLAID: Efficient Incremental Indexing via Monotonic Radius Hierarchical Partitioning for Multi-Vector Retrieval. PVLDB, 20(3): 899 - 911, 2026.
doi:10.14778/3860016.3860049

*Xujia Li is the corresponding author.
This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 20, No. 3 ISSN 2150-8097.
doi:10.14778/3860016.3860049

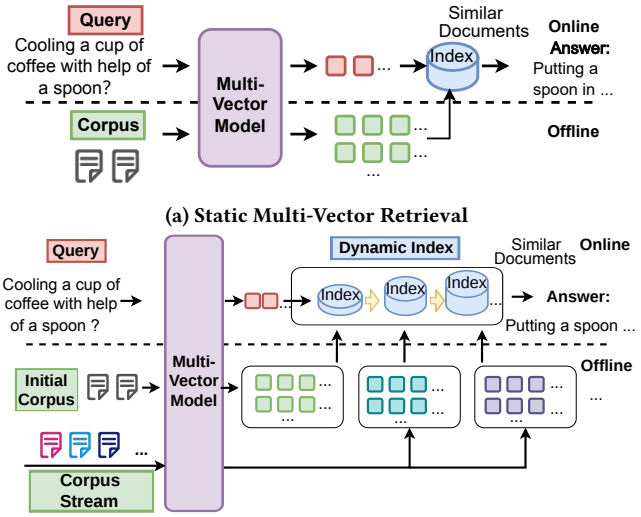


Figure 1: Multi-Vector Retrieval Framework

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Matthew202209/StreamPLAID.

1 INTRODUCTION

Retrieval-augmented generation (RAG) relies heavily on Multi-Vector Retrieval (MVR) to achieve state-of-the-art semantic matching, providing crucial infrastructural support for modern large language model (LLM) applications [9, 22, 25, 27, 29]. By encoding documents as granular sets of embeddings, as shown in Figure 1a, utilizing late-interaction MaxSim scoring [6, 17, 18, 32], MVR preserves fine-grained information far better than traditional Single-Vector Retrieval [6].

However, this superior retrieval precision comes at a steep cost: a massive memory footprint and prohibitive online retrieval latency. To mitigate these overheads, significant strides have been made to compress memory and reduce the online retrieval cost

by using Vector Quantization (VQ) based on Lloyd's algorithm coupled with Residual Quantization (RQ), like ColBERTv2 [18], PLAID [17], IGP [2]. While these techniques successfully resolve the efficiency bottlenecks for offline systems, their underlying mechanisms fundamentally assume a static document corpus.

As illustrated in Figure 1b, however, real-world retrieval systems operate in dynamic environments where documents are continuously streamed over time, such as high-frequency e-commerce catalog updates or shifting social media dynamics like the sudden emergence of COVID-19 topics. The state-of-the-art (SOTA) methods [2, 17, 18] based on Lloyd's algorithm [12, 14], expose severe limitations when confronted with dynamic situations.

(1) Lack of theoretical optimality: Lloyd's algorithm lacks theoretical guarantees for global optimality. Its high sensitivity to stochastic initialization easily traps the optimization in local minima, fundamentally degrading representation fidelity. In streaming environments, this inherent instability translates into severe retrieval performance fluctuations during continuous re-clustering, a phenomenon we empirically validate in Section 5.5 (**Figure 10d**).

(2) Prohibitive write amplification: The batch-oriented nature of Lloyd's algorithm fundamentally necessitates global re-clustering over the entire accumulated dataset to maintain index fidelity. This mechanism creates a prohibitive write amplification bottleneck, rendering continuous multi-vector updates computationally unscalable for real-time deployment.

(3) Vulnerability to concept drift: The naive "frozen-centroid" heuristic fundamentally fails under distribution shifts. Emerging embeddings representing novel topics naturally exhibit significantly different spatial distributions in the vector space (**Figure 2a**). Consequently, forcing these out-of-domain vectors to map onto distant, outdated centroids drastically increases quantization errors. This structural mismatch leads to a severe degradation in retrieval precision for SOTA methods relying on VQ and RQ, such as PLAID [17] (**Figure 2b**).

To systematically address these fundamental indexing bottlenecks, we propose **StreamPLAID**, an incremental indexing framework built upon PLAID [17], designed for efficient index maintenance in streaming MVR (Figure 5).

Specifically, to establish theoretical guarantees, we introduce **Monotonic Radius Hierarchical Partitioning (MRHP)**, transforming conventional flat clustering into a multi-layered architecture. By greedily assigning points to nearest centroids and cascading distant outliers, MRHP stratifies the data space by density, achieving a theoretically proven near-optimal centroid initialization. Leveraging this density stratification, we design an efficient **Streaming Routing** mechanism that eliminates prohibitive write amplification: incoming data in dense regions is routed to intermediate layers, while sparse data is directed to the final layer. Crucially, MRHP's structural properties ensure that localized repairing exclusively on this final layer is sufficient to maintain overall index stability. To manage concept drift without global reconstruction, we continuously monitor the final layer's dispersion against theoretical ideals. Once triggered, lightweight Clustering Features (CF) pinpoint degraded partitions, allowing StreamPLAID to perform efficient, localized repairs to update cluster centers.

The contributions of our framework are summarized as follows:

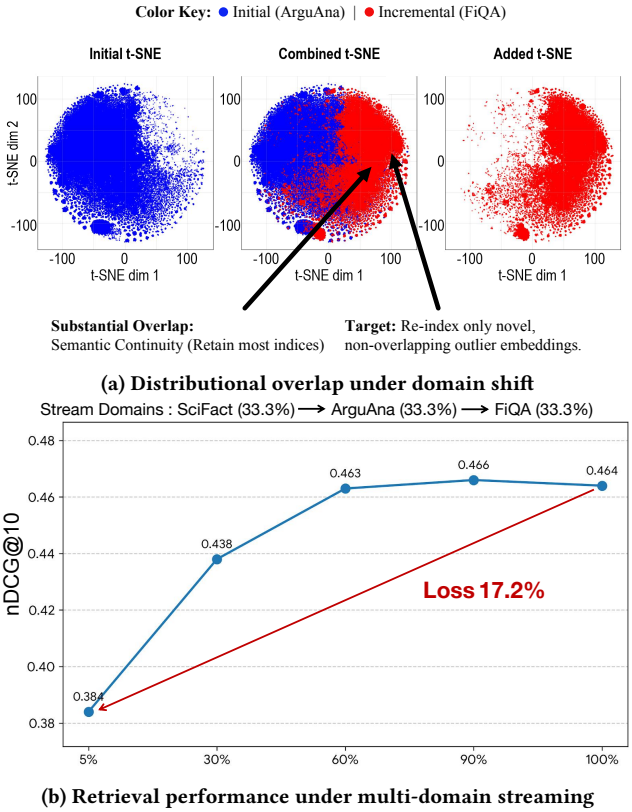


Figure 2: Impact of domain shift on retrieval and embedding distributions. (a) t-distributed Stochastic Neighbor Embedding (t-SNE) visualization of initial and incremental token embeddings. (b) Retrieval performance of PLAID [17] on a multi-domain stream with centroids initialized from the first $k\%$ of the data and kept fixed.

- **Novel Partition Index Structure:** We introduce MRHP, transitioning from conventional flat clustering to a topology-aware hierarchical peeling strategy. This systematic structure enables the framework to maintain a high-probability $\mathcal{O}(1)$-approximation guarantee for centroid optimality across both index initialization and incremental streaming updates.

- **Streaming Data Routing Strategy:** Leveraging MRHP's structure, we design a hierarchical routing mechanism that filters incoming data into distinct dense or sparse layers. By structurally isolating stable regions from unpredictable updates, this strategy circumvents global re-clustering and drastically mitigates write amplification.

- **Spatiotemporal Maintenance Operator:** Exploiting MRHP's bounded intermediate-layer errors, we design a dynamic maintenance operator that updates only the final layer. By comparing observed and expected errors to trigger updates, it utilizes lightweight Clustering Features (CF) to pinpoint and locally repair drift-affected partitions.

2 PRELIMINARIES

This section establishes the formal foundation for our proposed framework. As we primarily focus on the MVR task, we first introduce some important definitions to standardize our notations. For ease of reference, the major notations are summarized in Table 1.

Definition 2.1 (Multi-Vector Retrieval & MaxSim). Let a document and a query be represented as multi-vectors $\mathbf{D} = \{\mathbf{d}_1, \dots, \mathbf{d}_m\}$ and $\mathbf{Q} = \{\mathbf{q}_1, \dots, \mathbf{q}_n\}$ in $\mathbb{R}^{\text{Dim}}$. The *MVR* problem aims to retrieve the most relevant documents from a collection $\mathcal{D}$ according to the MaxSim score:

$$S(\mathbf{Q}, \mathbf{D}) = \sum_{i=1}^{n} \max_{1 \le j \le m} \text{sim}(\mathbf{q}_i, \mathbf{d}_j) \tag{1}$$

where $\text{sim}(\cdot, \cdot)$ is a vector similarity function [6].

To accelerate exact MVR, systems like ColBERTv2 [18] and PLAID [17] adopt a **Quantized Retrieval Paradigm**. Given a codebook of centroids $\mathcal{M}$, each vector $\mathbf{v}$ is compressed into its nearest centroid $\phi(\mathbf{v}) \in \mathcal{M}$ and a residual $\pi(\mathbf{v})$, yielding an approximate reconstruction $\tilde{\mathbf{v}} = \phi(\mathbf{v}) + \pi(\mathbf{v})$. This enables a highly efficient two-stage pipeline:

- **Candidate Generation:** Each query vector $\mathbf{q}_i$ matches proximate centroids M_{q_i}. An inverted index $\mathcal{I}$ is then used to efficiently fetch a candidate document set $\mathcal{D}_{\text{cand}} = \bigcup_{i=1}^{n} \bigcup_{\mu \in M_{q_i}} \mathcal{I}(\mu)$.
- **Residual Reconstruction & Scoring:** For each candidate $D \in \mathcal{D}_{\text{cand}}$, its multi-vector representation is dynamically reconstructed using the stored residuals to form $\tilde{D}$. The candidates are finally ranked using the approximate MaxSim score.

Definition 2.2 (Codebook Optimization Objective). Define $\mathcal{M} = \{\mu_1, \mu_2, \dots, \mu_k\}$ as the codebook consisting of k centroids. To ensure representational fidelity, we quantify the quality of $\mathcal{M}$ over a dataset $\mathcal{D}$ using the *Within-Cluster Sum of Squares (WCSS)*, denoted as $\mathcal{W}(\mathcal{D}, \mathcal{M})$. It measures the total quantization distortion and is defined as:

$$\mathcal{W}(\mathcal{D}, \mathcal{M}) = \sum_{\mathbf{d} \in \mathcal{D}} \min_{\mu \in \mathcal{M}} \|\mathbf{d} - \mu\|^2. \tag{2}$$

The goal of the indexing phase is to find an optimal codebook $\mathcal{M}^*$ that minimizes the total $\mathcal{W}(\mathcal{D}, \mathcal{M})$ (WCSS). For brevity, we use W to denote $\mathcal{W}(\mathcal{D}, \mathcal{M})$ when the underlying dataset and codebook are clear from the context.

Definition 2.3 (Streaming Codebook Optimization). Let the data stream be a sequence of batches $\mathcal{S} = \{\mathcal{D}_0, \mathcal{D}_1, \dots, \mathcal{D}_t, \dots\}$, where $\mathcal{D}_t$ represents the set of incoming token embeddings at time step t. Consequently, the cumulative corpus available at time T, denoted as $\mathcal{V}_T$, is defined as the union of the initial dataset $\mathcal{D}_0$ and the historical stream sequence:

$$\mathcal{V}_T = \mathcal{D}_0 \cup \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_T. \tag{3}$$

The objective of streaming indexing is to maintain a dynamic codebook $\mathcal{M}_T^*$ that minimizes the total quantization error over the entire cumulative corpus $\mathcal{V}_T$. Formally, the ideal optimization objective is defined as:

$$\mathcal{M}_T^* = \arg\min_{\mathcal{M}} \mathcal{W}(\mathcal{V}_T, \mathcal{M}) \tag{4}$$

We abstract the index maintenance task in streaming multi-vector retrieval as a **Dynamic Voronoi Partitioning** problem. The core objective is to design an efficient online algorithm that incrementally updates the codebook $\mathcal{M}_{t-1} \to \mathcal{M}_t$ upon observing new data $\mathcal{D}_t$, achieving a theoretically guaranteed approximation of the global optimum without revisiting historical raw data.

Definition 2.4 (Cost-Driven Maintenance Criterion [16]). By normalizing $\mathcal{W}$(WCSS) against the corresponding data cardinality N, we define the observed **mean quantization error** as $E = \mathcal{W}/N$. Let E' denote the theoretical ideal error if the data were optimally re-clustered. We define the **Optimality Gap** G to monitor structural degradation:

$$G = \frac{|E - E'|}{E'}, \quad \text{with} \quad E' = E^{(0)} \left(\frac{N_0}{N}\right)^{1/\text{Dim}} \tag{5}$$

where $E^{(0)}$ and N_0 denote the baseline error and cardinality at the last structural update, and Dim is the vector dimensionality. A structural repair is triggered when $G \ge \tau$, where τ **is the predefined system tolerance threshold.**

Table 1: Key Notations

Symbol	Description
$\mathbf{Q}, \mathbf{D}$	Query and document multi-vectors
$\mathbf{q}_i, \mathbf{d}_j$	Individual token embeddings ($\in \mathbb{R}^{\text{Dim}}$)
$\mathcal{M}, \mu, k_l, K$	Codebook (cluster centroids), individual centroid, number of centroids in layer l, and total number of centroids across all layers
$\phi(\cdot), \pi(\cdot)$	Centroid mapping function and quantized residual
$\tilde{v}_i, \tilde{\mathbf{D}}$	Approximate reconstructed vector and document
$\mathcal{W}, N$	Within-Cluster Sum of Squares (WCSS) and cardinality
E, E'	Observed and theoretical ideal mean quantization error
G, τ	Optimality gap and predefined system tolerance threshold
L_{max}, r_l	Number of layers and maximum routing radius for layer l

3 PLAID REVISITED

In this section, we revisit the conventional paradigm of state-of-the-art MVR approaches, primarily based on the PLAID framework [2, 3, 10, 11, 13, 17]. We first detail its fundamental construction and retrieval pipeline, and subsequently analyze its severe performance bottlenecks in dynamic streaming scenarios.

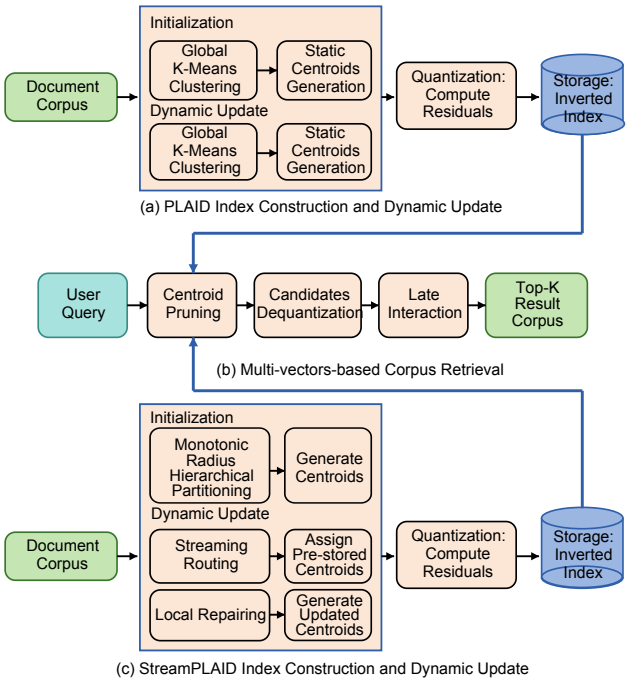


Figure 3: The index construction and retrieval workflows. (a) PLAID Index Construction and Dynamic Update; (b) Multi-vectors-based Corpus Retrieval; (c) StreamPLAID Index Construction and Dynamic Update.

3.1 PLAID Index Construction and Retrieval Process

Index Construction: As illustrated in Figure 3, the offline construction involves three phases. First, the clustering phase generates global centroids and assigns tokens to their nearest centroids. Second, the RQ phase computes the residual vector for each token relative to its assigned centroid. Finally, the inverted index construction phase maps centroid IDs to the corresponding document IDs within each cluster.

Retrieval Process: To accelerate queries and bypass exhaustive late-interaction computation, PLAID capitalizes on its index structure for efficient candidate screening. By treating centroids as reliable proxies for full passage embeddings, it safely discards low-scoring candidates without the computational overhead of decompressing their residuals [17]. Finally, the system decompresses the residuals exclusively for this refined subset of candidates, executing the full late-interaction scoring to determine the ultimate ranking.

3.2 Dynamic Scenarios

PLAID builds a centroid-based index by clustering document vectors into k centroids and using these centroids for candidate screening. While effective for static datasets, this design faces two challenges in streaming environments. First, concept drift can degrade centroid quality by increasing quantization error

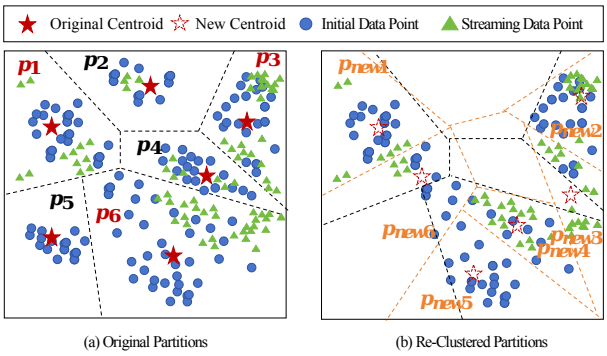


Figure 4: An example of the centroid drift issue. Flat re-clustering drags new centroids (c_{new}) into intermediate spaces, causing p_{new3} to overlap heavily with p_4. This spatial ambiguity misdirects query q to p_{new3} since $dist(c_{p_{new3}}, q) < dist(c_{p_{new4}}, q)$, missing the true targets in p_4.

and dispersing vectors within clusters, thereby reducing screening effectiveness (Figure 2a). Second, restoring centroid quality through conventional rebuilding requires full-data reclustering. Since PLAID sets k to the largest power of two not exceeding $16\sqrt{N}$, we have $k = \Theta(\sqrt{N})$. Thus, clustering all N vectors incurs $\mathcal{O}(Nk) = \mathcal{O}(N\sqrt{N})$ computational cost, while updating indexed representations introduces substantial write amplification, making frequent rebuilding expensive under continuous data ingestion.

4 METHODOLOGY

Figure 5 provides an overview of StreamPLAID. Sections 4.1–4.3 present MRHP construction, Streaming Routing, and Localized Repairing, respectively, while Sections 4.4–4.6 provide the corresponding theoretical and efficiency analyses.

4.1 Monotonic Radius Hierarchical Partitioning

To enhance the update efficiency of MVR in streaming data scenarios, we abandon the practice of full index reconstruction and instead adopt a localized update strategy. Inspired by Figure 2a, the most straightforward approach is to directly assign incoming data points to their nearest partitions, followed by localized re-clustering for any partitions that exhibit significant data deviation.

However, applying local re-clustering directly to a flat partition architecture is fundamentally flawed due to the **Centroid Drift Issue** (see Figure 4). Because flat partitions ignore density disparities, re-clustering streaming outliers with a dense core forcefully drags the centroid into empty space. This drift inevitably causes spatial overlap between clusters, leading to severe routing ambiguity and degraded retrieval accuracy.

Notably, Mettu and Plaxton [15] introduced the Successive Sampling strategy, demonstrating that an $\mathcal{O}(1)$-approximation for optimal clustering can be guaranteed with high probability through a greedy, layer-wise peeling of data points in close proximity to the cluster centers. On a different axis, TaCo [24] shows that adapting

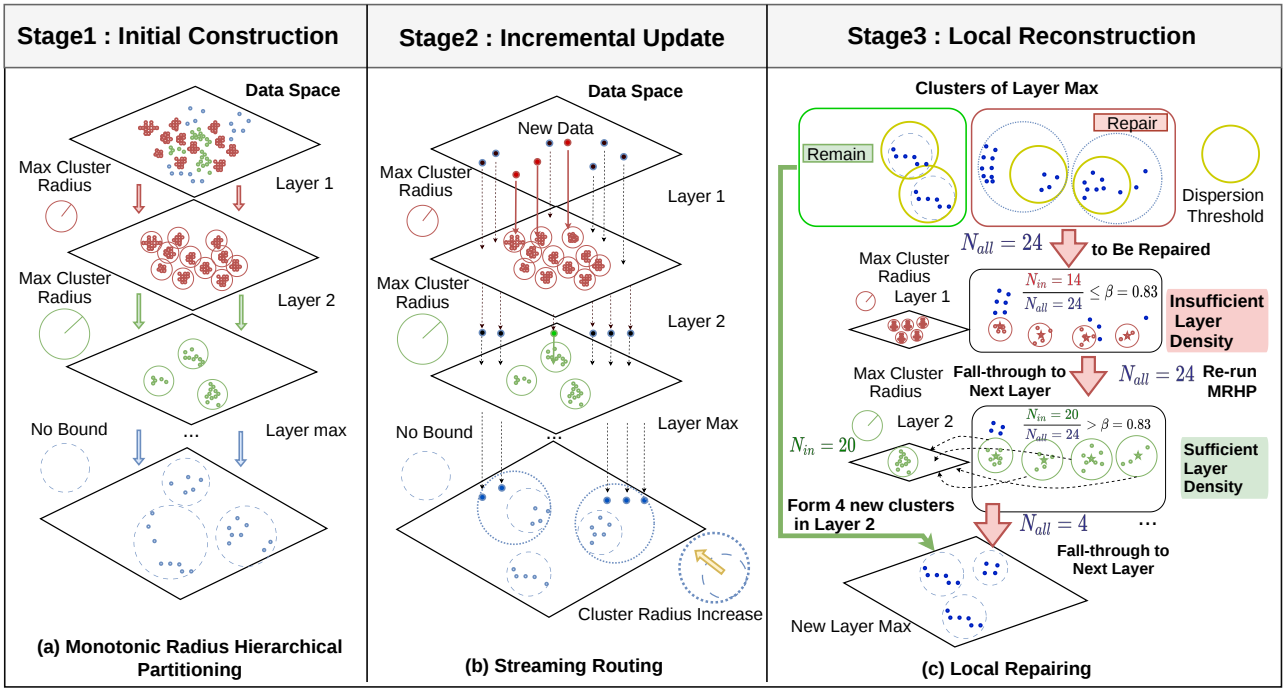


Figure 5: Overview of the StreamPLAID framework. (a) Monotonic Radius Hierarchical Partitioning: MRHP stratifies the initial dataset by greedily assigning points to bounded nearest centroids and cascading distant outliers to deeper layers. (b) Streaming Routing: The streaming routing mechanism routes bounded incoming data to intermediate dense layers and directs unbounded sparse outliers to the final layer. (c) Local Repairing: MRHP is locally re-run on dispersed clusters within the final layer. New clusters are recursively assigned to appropriate layers based on the initial retention ratio β, with remaining outliers falling through to form a new final layer.

Algorithm 1 Monotonic Radius Hierarchical Partitioning

Require: Dataset $\mathcal{U}$, number of layers $L_{\max}$, retention ratio β
Ensure: Hierarchical index $\mathcal{H}$

```
l ← 1, U_1 ← U, r_0 ← 0, H ← [ ]
while l < L_max and U_l ≠ ∅ do
    (M_l, S_l, r_l) ← BuildLayer(U_l, β, r_{l−1}, l, L_max)
    Append(H, (M_l, S_l, r_l))
    U_{l+1} ← U_l \ S_l
    l ← l + 1
end while
if U_l ≠ ∅ then
    (M_l, S_l, r_l) ← BuildLayer(U_l, β, r_{l−1}, l, L_max)
    Append(H, (M_l, S_l, r_l))
end if
return H
```

the index to the intrinsic data distribution is crucial for efficient high-dimensional ANN search.

Motivated by these two observations, we introduce a novel hierarchical partition method named MRHP. MRHP integrates the core concept of Successive Sampling. Using a pre-defined retention ratio $\beta \in (0, 1)$, it iteratively discovers centroids and greedily retains at least a β fraction of nearby data points at each layer, thereby preserving an $\mathcal{O}(1)$-approximation to the optimal clustering cost with high probability (detailed proofs are deferred to Section 4.4). To ensure centroid continuity and preserve theoretical bounds, we replace the standard random sampling approach with an **initialization-and-refinement strategy**. Specifically, in each layer, we first randomly sample the target number of points to serve as initial centroids, and subsequently execute a clustering process to refine their positions over the unassigned data. Algorithms 1 and 2 formalize the overall MRHP procedure and the layer-wise construction, respectively.

Specifically, the strategy is executed independently at each layer. For a given layer l, the algorithm processes the subset of unassigned data points $\mathcal{U}_l$ cascading from the preceding layer. We first allocate a layer-specific centroid count k_l (Algorithm 2, Line 1), with fewer centroids assigned to deeper layers as the residual data progressively decreases. Let $K = \sum_{l=1}^{L_{\max}} k_l$ denote the total centroid budget across all MRHP layers, with implementation details provided in Section 5.1.1. We then uniformly sample k_l points from $\mathcal{U}_l$ as the initial cluster centers (Algorithm 2, Line 2), followed by K-Means refinement over $\mathcal{U}_l$ to obtain the final centroid set $\mathcal{M}_l$ for the current layer (Algorithm 2, Line 3).

Once the refined centroids are established, we compute the distance from each point in $\mathcal{U}_l$ to its nearest centroid. The candidate radius r_{cand} is defined as the minimum radius that covers at least

Algorithm 2 BuildLayer

Require: Points entering this layer $\mathcal{U}_l$, retention ratio β, previous-layer radius r_{l-1}, current layer l, number of layers $L_{\max}$
Ensure: Centroid set $\mathcal{M}_l$, covered point set S_l, layer radius r_l
1: Allocate centroid count k_l for layer l
2: $\mathcal{M}_l^{\text{init}} \leftarrow \text{RandomSample}(\mathcal{U}_l, k_l)$
3: $\mathcal{M}_l \leftarrow \text{Cluster}(\mathcal{U}_l, \mathcal{M}_l^{\text{init}})$
4: **if** $l = L_{\max}$ **then**
5: $\quad r_l \leftarrow +\infty$
6: **else**
7: $\quad r_{\text{cand}} \leftarrow \min\left\{ r \geq 0 \;\middle|\; \frac{\left|\{x \in \mathcal{U}_l : \min_{\mu \in \mathcal{M}_l} \|x-\mu\| \leq r\}\right|}{|\mathcal{U}_l|} \geq \beta \right\}$
8: $\quad r_l \leftarrow \max(r_{\text{cand}}, r_{l-1})$
9: **end if**
10: **for all** $\mu \in \mathcal{M}_l$ **do**
11: $\quad S_{l,\mu} \leftarrow \{x \in \mathcal{U}_l \mid \mu = \arg\min_{z \in \mathcal{M}_l} \|x - z\| \wedge \|x - \mu\| \leq r_l\}$
12: **end for**
13: $S_l \leftarrow \bigcup_{\mu \in \mathcal{M}_l} S_{l,\mu}$
14: **return** $(\mathcal{M}_l, S_l, r_l)$

a β fraction of the points in $\mathcal{U}_l$ (Algorithm 2, Line 7). We then enforce a monotonic non-decreasing radius constraint across successive layers by setting $r_l = \max(r_{\text{cand}}, r_{l-1})$, which guarantees $r_1 \leq r_2 \leq \cdots \leq r_{L_{\max}-1}$. This produces a density-aware hierarchy in which earlier layers capture dense regions under tighter radius constraints, while deeper layers progressively accommodate sparser residual regions.

Given the resulting radius r_l, each point is assigned to its nearest centroid if it lies within the radius constraint. Specifically, $S_{l,\mu}$ denotes the partition associated with centroid $\mu \in \mathcal{M}_l$, containing all points assigned to μ whose distances do not exceed r_l. The covered set of layer l is then defined as $S_l = \bigcup_{\mu \in \mathcal{M}_l} S_{l,\mu}$. The remaining unassigned points, $\mathcal{U}_{l+1} = \mathcal{U}_l \setminus S_l$, cascade to the next layer and undergo the same clustering and peeling procedure.

This successive peeling process continues until the final layer $l = L_{\max}$. At this terminal layer, the radius constraint is completely relaxed by setting $r_{L_{\max}} = +\infty$, so that all remaining sparse, long-tail residual points are assigned to their nearest centroids and absorbed into the final layer.

4.2 Incremental Update by Streaming Routing

A fundamental limitation of static MVR approaches is that they are spatially agnostic. From this perspective, all Voronoi partitions are treated as **spatially equivalent**, ignoring the inherent non-uniformity of the data distribution. Consequently, these methods lack the mechanism to differentiate between stable dense regions and drifting sparse areas.

However, our empirical analysis demonstrates that a large proportion of tokens from the new corpus still occupy the high-density regions of the original vector space despite the significant content shift shown in Figure 2a. This suggests that the semantic core of the data remains relatively stable.

Leveraging this observation, we propose an adaptive **Streaming Routing** strategy based on the MRHP structure. Unlike conventional partitioning, our hierarchical design allows us to adaptively distinguish between incoming data points that fall into established high-density regions and those belonging to the sparse long-tail (Figure 5b). Specifically, the routing process operates as a **Filter-and-Leak** mechanism. Upon the arrival of a new document token vector d, the system traverses the MRHP hierarchy starting from the first layer ($l = 1$). For each layer l, we compute the distance from d to its nearest centroid in $\mathcal{M}_l$. Depending on this distance, the token is handled in three cases as follows:

- **Capture (Dense Region):** If $\min_{\mu \in \mathcal{M}_l} \|d - \mu\| \leq r_l$, assign d to the nearest-centroid partition.
- **Leak (Sparse Region):** If $\min_{\mu \in \mathcal{M}_l} \|d-\mu\| > r_l$, the token "leaks" to the subsequent layer ($l + 1$).
- **Fallback (Tail Handling):** This process repeats until the token is captured or reaches the final layer ($L_{\max}$), where it is assigned to the nearest centroid unconditionally.

4.3 Topology-Aware Subset Extraction and Localized Repair

In a streaming scenario, spatial deterioration in the final sparse layer is often localized due to concentrated outlier tokens or topic-specific concept drift. When the final-layer Optimality Gap $G_{L_{\max}}$, formally defined in Section 4.5, reaches its threshold $\tau_{L_{\max}}$, i.e., $G_{L_{\max}} \geq \tau_{L_{\max}}$, a localized repair phase is initiated.

The critical challenge during this phase is *how* to efficiently extract the exact degraded subset ($\mathcal{U}_{sub}$) prior to local reconstruction, without scanning the raw data of stable clusters. To achieve this, we introduce a micro-level, topology-aware extraction mechanism.

By adopting the Clustering Features (CF) concept [30], we can efficiently pinpoint which final-layer partitions require reconstruction. For each partition $S_{L_{max},\mu}$, as defined in Algorithm 2, in the final layer, we maintain a dynamic summary tuple $CF = (N_{L_{\max},\mu}, LS_{L_{\max},\mu}, SS_{L_{\max},\mu})$, defined as follows:

- $N_{L_{\max},\mu}$: the total number of vectors in the cluster;
- $LS_{L_{\max},\mu} = \sum_{n=1}^{N_{L_{\max},\mu}} d_n$: the linear sum of the cluster vectors;
- $SS_{L_{\max},\mu} = \sum_{n=1}^{N_{L_{\max},\mu}} \|d_n\|^2$: the sum of squared norms.

Leveraging the additivity of these sufficient statistics, we can update the cluster profile in $O(1)$ time upon the arrival of new tokens. To quantify the cluster compactness, we define the Local Dispersion Metric $\Delta(S_{L_{max},\mu})$, which represents the root mean square deviation of the vectors from the cluster centroid:

$$\Delta(S_{L_{max},\mu}) = \sqrt{\frac{SS_{L_{\max},\mu}}{N_{L_{\max},\mu}} - \left\| \frac{LS_{L_{\max},\mu}}{N_{L_{\max},\mu}} \right\|^2} \tag{6}$$

Mathematically, Eq. (6) is equivalent to the root of the trace of the cluster's covariance matrix. It provides a robust measure of the spatial spread of vectors within $S_{L_{max},\mu}$. Since $\Delta(S_{L_{max},\mu})$ relies solely on summary statistics, it allows for highly efficient spatial monitoring.

Subset Aggregation and Reconstruction: During the extraction phase, we evaluate $\Delta(S_{L_{max},\mu})$ for all partitions in the final

layer. Any partition where $\Delta(S_{L_{max},\mu})$ exceeds a predefined local dispersion threshold (τ_Δ) is flagged as degraded. The subset $\mathcal{U}_{sub}$ is then rigorously defined as the union of all vectors residing within these flagged partitions. As illustrated in Figure 5c, the MRHP algorithm is subsequently re-executed exclusively on $\mathcal{U}_{sub}$ to form updated clusters, restoring the index quality while keeping the stable partitions completely untouched.

Crucially, during this local reconstruction, the maximum cluster radius r_l for each intermediate layer is strictly fixed to its initially constructed value. For each layer in the hierarchy, we evaluate whether the clusters formed under this fixed radius can capture a sufficient proportion of the candidate points to meet the predefined retention ratio β. If the layer density is insufficient (the captured ratio is less than β), the layer is deemed too sparse for this specific subset, and all candidate points completely fall through to the next layer without forming any clusters. Conversely, if the density requirement is met, the captured points successfully form new clusters and remain in the current layer, while only the unassigned residual points cascade downward. This topology-aware rerouting dynamically reallocates the degraded vectors into appropriate hierarchical tiers based on their updated spatial distributions.

4.4 High-Probability Constant-Factor Approximation Guarantee

In this section, we establish the approximation guarantee of MRHP and show that its initialization procedure provides a constant-factor approximation invariant for subsequent streaming maintenance. Based on this invariant, we further demonstrate that the proposed routing and reconstruction mechanisms preserve the approximation guarantee throughout the StreamPLAID lifecycle. We denote OPT_K as the clustering cost of the globally optimal K-means solution over dataset $\mathcal{U}$, where K denotes the total centroid budget of MRHP.

Theorem 4.1 (High-Probability Approximation Guarantee of Static MRHP). *For any dataset $\mathcal{U}$, the center set $\mathcal{M}_{mrhp}$ generated by the MRHP algorithm achieves an $\mathcal{O}(1)$-approximation to the optimal clustering cost with high probability.*

Proof Sketch. MRHP follows the Successive Sampling framework proposed by Mettu and Plaxton [15]. At each hierarchical layer, the algorithm first randomly samples representative centers according to the Successive Sampling strategy. Let $\mathcal{M}_{\text{init}}$ denote the aggregate set of layer-wise initial centroids prior to their respective Lloyd refinements. According to the approximation analysis of Successive Sampling, the obtained initialization achieves a constant-factor approximation to the optimal clustering objective with high probability ($\mathcal{W}(\mathcal{U}, \mathcal{M}_{init}) \leq \mathcal{O}(1)OPT_K$). After initialization, MRHP performs Lloyd refinement on the sampled centers at each layer. Since Lloyd's algorithm monotonically decreases the WCSS objective $\mathcal{W}$ [12, 14], the refinement procedure only reduces the clustering cost ($\mathcal{W}(\mathcal{U}, \mathcal{M}_{mrhp}) \leq \mathcal{W}(\mathcal{U}, \mathcal{M}_{init})$). Therefore, the layer-wise refinement preserves the approximation guarantee inherited from Successive Sampling. Aggregating all layers gives the final center set $\mathcal{M}_{mrhp}$ satisfying $\mathcal{W}(\mathcal{U}, \mathcal{M}_{mrhp}) \leq \mathcal{O}(1)OPT_K$ with high probability. □

Remark 1 (Practical Implementation via Mini-Batch). While Theorem 4.1 assumes exact Lloyd's algorithm for theoretical analysis, exact iterative refinement is computationally expensive for large-scale vector retrieval systems. Therefore, MRHP adopts Mini-Batch K-Means [19] in practical implementations. Although Mini-Batch K-Means relaxes the deterministic monotonic non-increasing property of standard Lloyd refinement, sufficiently large batch sizes provide stable convergence behavior and effectively approximate the refinement process. This practical optimization accelerates index construction while empirically maintaining similar clustering performance.

Theorem 4.2 (Approximation Preservation under Incremental Updates). *Conditioned on the successful initialization event of MRHP, if the index satisfies an $\mathcal{O}(1)$-approximation guarantee with high probability at time step t, the proposed maintenance strategy preserves the same approximation order after incremental vector insertion or triggered local reconstruction.*

Proof Sketch. We analyze the two streaming maintenance mechanisms. **For incremental update by streaming routing**, when a new vector d_t is inserted into an intermediate layer, the routing mechanism assigns it to the nearest centroid satisfying the layer radius constraint $\|d_t - \mu\|^2 \leq r_l^2$. Since the intermediate layer radius is inherited from the approximation-preserving initialization procedure, the additional quantization error introduced by insertion remains locally bounded. For vectors assigned to the final sparse layer $L_{\max}$, the accumulated residual error is monitored by the reconstruction threshold, and local reconstruction is triggered once the error exceeds the predefined tolerance to prevent unbounded approximation degradation. **For local repairing**, Static MRHP is re-executed on the affected subset $\mathcal{U}_{sub}$. According to Theorem 4.1, the reconstructed local index achieves an $\mathcal{O}(1)$-approximation guarantee with high probability. Since the clustering objective is additive over disjoint partitions, replacing the affected subset with the reconstructed sub-index restores local approximation quality without increasing the asymptotic global approximation bound. □

Consequently, approximation-preserving initialization, bounded-radius routing, and adaptive local reconstruction jointly enable **StreamPLAID** to maintain a high-probability $\mathcal{O}(1)$-approximation guarantee throughout its lifecycle.

4.5 Error-Bounded Cost Model and Selective Maintenance Strategy

We restrict structural re-clustering to the final sparse layer. We show that intermediate-layer errors remain bounded by a time-invariant radius-based bound, whereas the final-layer error may accumulate over time.

To formalize this analysis, we recall the mean quantization error $E = \mathcal{W}/N$. For an MRHP index with $L_{\max}$ layers, let N_l and $\mathcal{W}_l$ denote the cardinality and WCSS of layer l, respectively, yielding the locally normalized error $E_l = \mathcal{W}_l/N_l$. We logically partition the global WCSS into the intermediate layers and the final sparse layer as $\mathcal{W}_{\text{total}} = \sum_{l=1}^{L_{\max}-1} \mathcal{W}_l + \mathcal{W}_{L_{\max}}$. Correspondingly, after normalization by the global cardinality N_{total}, we define the globally normalized contributions of the intermediate and final layers

as $\bar{E}_{\text{inter}} = \sum_{l=1}^{L_{\max}-1} \mathcal{W}_l / N_{\text{total}}$ and $\bar{E}_{L_{\max}} = \mathcal{W}_{L_{\max}} / N_{\text{total}}$, respectively, such that $E_{\text{total}} = \bar{E}_{\text{inter}} + \bar{E}_{L_{\max}}$.

Lemma 4.3 (Radius-Bounded Intermediate Layers). *For every intermediate layer $l \in [1, L_{max} - 1]$, the quantization error satisfies $E_l \le r_l^2$, where r_l is the predefined maximum routing radius.*

Proof Sketch. Because every vector x assigned to layer l is constrained by its routing centroid μ_x such that $\|x - \mu_x\|^2 \le r_l^2$, its average quantization error $E_l = \mathcal{W}_l / N_l$ is also directly upper-bounded by r_l^2. □

Theorem 4.4 (Time-Invariant Intermediate Error Bound). *Define the globally normalized intermediate error contribution as*

$$\bar{E}_{inter} = \sum_{l=1}^{L_{max}-1} \frac{\mathcal{W}_l}{N_{total}}. \tag{7}$$

Then $\bar{E}_{inter}$ is upper-bounded by a time-invariant constant C.

Proof Sketch. Since $N_{total} \ge N_l$ for any layer l, we can perform an upward relaxation on the denominator and apply Lemma 4.3:

$$\begin{aligned} \bar{E}_{inter} &= \sum_{l=1}^{L_{max}-1} \frac{\mathcal{W}_l}{N_{total}} \le \sum_{l=1}^{L_{max}-1} \frac{\mathcal{W}_l}{N_l} \\ &= \sum_{l=1}^{L_{max}-1} E_l \le \sum_{l=1}^{L_{max}-1} r_l^2 = C. \end{aligned} \tag{8}$$

Since all routing radii are fixed after index construction, C is independent of stream length, insertion time, and database cardinality. □

Corollary 4.5 (Global Error Decomposition). *The global mean quantization error exactly satisfies $E_{\text{total}} = \bar{E}_{\text{inter}} + \bar{E}_{L_{\max}}$. Because $\bar{E}_{\text{inter}} \le C$, only the final sparse layer can contribute unbounded global degradation.*

Proof. Since $\mathcal{W}_{\text{total}} = \sum_{l=1}^{L_{max}-1} \mathcal{W}_l + \mathcal{W}_{L_{max}}$, dividing both sides by N_{total} gives $E_{\text{total}} = \bar{E}_{\text{inter}} + \bar{E}_{L_{\max}}$. The second statement follows directly from Theorem 4.4. □

Dynamic Analysis & Maintenance Rule. Consider two arbitrary timestamps $T_1 < T_2$ during streaming insertion. The variation of the global error is $\Delta E_{\text{total}} = E_{\text{total}}(T_2) - E_{\text{total}}(T_1)$. Substituting the decomposition from Corollary 4.5, we obtain $\Delta E_{\text{total}} = \Delta \bar{E}_{\text{inter}} + \Delta \bar{E}_{L_{\max}}$. Since $\bar{E}_{\text{inter}} \le C$, intermediate-layer fluctuations cannot cause unbounded long-term degradation; hence, sustained global degradation can only arise from $\bar{E}_{L_{\max}}$.

Following the cost-driven maintenance criterion in Definition 2.4, we apply the same criterion to the final sparse layer. Its locally normalized quantization error is $E_{L_{\max}} = \mathcal{W}_{L_{\max}} / N_{L_{\max}}$, and $E'_{L_{\max}}$ denotes the corresponding theoretical ideal error computed according to Definition 2.4. We then define the localized Optimality Gap as $G_{L_{\max}} = |E_{L_{\max}} - E'_{L_{\max}}| / E'_{L_{\max}}$. **Thus, monitoring its Optimality Gap suffices, triggering localized re-clustering if and only if $G_{L_{\max}} \ge \tau_{L_{\max}}$.**

4.6 Efficiency Analysis

We analyze the computational efficiency of Streaming Routing and Localized Repair, emphasizing that their costs do not scale with the continuously growing global corpus size N_t at time step t.

Streaming Routing: For an incoming vector of dimension *Dim*, let L_{route} denote the number of visited layers, with $\mathbb{E}[L_{\text{route}}] \ll L_{\max}$. Since $k_{l+1} \le k_l$, the expected routing cost is bounded by $\mathcal{O}(\mathbb{E}[L_{\text{route}}] \cdot k_1 \cdot Dim)$, while updating the Clustering Features incurs $\mathcal{O}(1)$ overhead. Because the layer-wise centroid counts are fixed after index construction, both k_1 and $\mathbb{E}[L_{\text{route}}]$ do not scale with N_t. Therefore, the per-vector routing cost is $\mathcal{O}(1)$ with respect to N_t.

Localized Repairing: Conventional global re-clustering methods, such as PLAID [17], require $\mathcal{O}(N_t\sqrt{N_t})$ time for global index maintenance. In contrast, our repair operation is restricted to a drifted subset of size N_{sub} with k_{sub} localized centroids, resulting in a repair cost of $\mathcal{O}(N_{\text{sub}} k_{\text{sub}})$. Under the localized-repair setting, where N_{sub} and k_{sub} remain bounded independently of the continuously growing global corpus size N_t, this cost does not scale with N_t and is therefore $\mathcal{O}(1)$ with respect to N_t.

In summary, **under bounded localized repair, our proposed method reduces the per-update index maintenance complexity from $\mathcal{O}(N_t\sqrt{N_t})$ to $\mathcal{O}(1)$ with respect to N_t.**

5 EXPERIMENTS

Table 2: Dataset statistics for continuous streaming adaptation.

Dataset	Init. Docs	Init. Qs	Added Docs/Step	Added Qs/Step
In-Domain Streams				
SciFact	2,591	150	~518	~30
ArguAna	4,336	703	~867	~141
FiQA	28,800	324	~5,760	~65
Touche2020	190,760	24	~38,152	~5
LoTTE-Pooled	1,409,551	1,934	~281,910	~387
Out-of-Domain Streams (Sequential Shifts)				
BEIR-OOD	5,183	300	5,183 (fixed)	{1406, 648, 49, 6666}
LoTTE-OOD	119,461	1,999	119,461 (fixed)	{924, 617, 596, 1070}

BEIR-OOD: SciFact (Init) → ArguAna → FiQA → Touche2020 → Fever.
LoTTE-OOD: Lifestyle (Init) → Recreation → Science → Technology → Writing.

We evaluate StreamPLAID against state-of-the-art MVR baselines across five in-domain and two out-of-domain streaming datasets. Focusing on index maintenance efficiency, retrieval accuracy, and system robustness under continuous updates, we aim to answer the following research questions (RQs):

RQ1: Can StreamPLAID achieve superior index maintenance efficiency under a streaming setting without compromising immediate retrieval effectiveness and long-term structural stability?

RQ2: What is the impact of re-clustering updates on retrieval accuracy under distribution shifts, and can StreamPLAID maintain high retrieval performance?

RQ3: How do different components of our StreamPLAID contribute to the retrieval performance?

RQ4: How robust is StreamPLAID under continuous data streams, and can it achieve highly efficient index maintenance without compromising retrieval accuracy?

Table 3: Average total index update time (s) across ID (Steps 1–5) and OOD (Steps 1–4). Values represent the mean time over the entire incremental update process. Bold indicates the best performance. StreamPLAID shows a clear efficiency advantage.

Method	In-Domain Datasets					Out-of-Domain Datasets	
	SciFact	ArguAna	FiQA	Touche2020	LoTTE-Pooled	BEIR-OOD	LoTTE-OOD
Full-Rebuild Baselines							
Dessert	8.25	15.21	220.61	2585.80	19050.12	220.49	1578.86
IGP	16.47	26.78	201.51	1667.12	17865.58	90.43	1234.81
PLAID	12.72	20.17	169.07	638.79	17113.56	55.87	985.99
XTR	18.54	26.43	113.60	751.11	4586.02	56.70	363.53
Incremental Update Methods							
MUVERA	24.08	47.25	334.57	1460.04	/	160.21	2919.14
StreamPLAID (Ours)	**0.73**	**0.91**	**3.66**	**30.74**	**512.28**	**8.56**	**205.47**

* "/" indicates missing data.

5.1 Experimental Settings

5.1.1 Implementation Details. StreamPLAID extends PLAID [17] by substituting its standard partition index with our MRHP index. To ensure reproducibility, our core configurations are detailed as follows:

- **Architecture Mechanics:** For incremental updates, we seamlessly combine Streaming Routing (Section 4.2) with Topology-Aware Localized Repairing (Section 4.3)[1]. The multi-vector retrieval process strictly follows PLAID's original logic (detailed in Section 3) to ensure a fair baseline comparison.
- **Hardware & Embeddings:** To evaluate pure algorithmic efficiency, all indexing and retrieval operations are executed exclusively on a **96-core Intel Xeon Platinum 8468 CPU** with **2 TB RAM**. Texts are encoded via ColBERTv2 ($Dim = 128$), utilizing an **NVIDIA H100 GPU (80GB VRAM)** strictly for embedding generation.
- **Hyperparameters:** Following PLAID [17], for both initial construction and local re-clustering, we set the global centroid budget K to the largest power of two not exceeding $16\sqrt{N}$, where N denotes the number of vectors to be clustered. We then distribute K across layers to obtain the layer-specific centroid counts k_l, using weights proportional to $(1-\beta)^{l-1}$. For MRHP, we set the layer count $L_{max} = \mathbf{3}$ and the retention ratio $\beta = \mathbf{0.83}$. For structural maintenance, we set the localized repair threshold $\tau_{L_{max}} = \mathbf{0.01}$ and the local dispersion threshold $\tau_{\Delta} = \mathbf{0.85}$.

5.1.2 Datasets and Incremental Scenarios Adaptation. Since mainstream multi-vector retrieval benchmarks lack dedicated datasets for dynamic environments, we construct realistic continuous data streams by adapting static corpora from the widely adopted BEIR [21] and LoTTE [18] benchmarks. As detailed in **Table 2**, we outline the exact corpus statistics (total numbers of documents, initial sizes, batch sizes) to simulate three distinct streaming scenarios:

- **In-Domain (ID) Incremental Stream (for RQ1):** Using BEIR (SciFact, ArguAna, FiQA, Touche2020) and LoTTE-Pooled, we simulate steady growth by initializing the index with 50% of the corpus and appending the rest across five 10% increments.
- **Out-of-Domain (OOD) Incremental Stream (for RQ2 & RQ3):** To evaluate robustness against extreme distribution shifts, we construct two aggregated datasets (BEIR-OOD and LoTTE-OOD) by combining distinct domains from their respective benchmarks. We initialize with the smallest domain and sequentially insert four distinct domains over four steps, ensuring each batch volume matches the initial set.
- **Continuous Streaming (for RQ4):** Simulating high-frequency updates, we initialize with 100 documents and execute 1,000 continuous updates at insertion rates of 10, 20, and 30 documents/step, sequentially transitioning across LoTTE subdomains[2]

5.1.3 Baselines. We evaluate our proposed approach by comparing it against various competitive baselines.

State-of-the-art static MVR methods include: 1. **PLAID** [17]: Leverages centroid-based clustering and token pruning to accelerate late interaction retrieval. 2. **IGP** [2]: Utilizes an incremental greedy probing strategy over proximity graphs to efficiently generate candidate sets. 3. **XTR** [7]: Retrieves only critical query-specific tokens and imputes missing similarities to optimize search. 4. **MUVERA** [4]: Transforms multi-vector sets into fixed-dimensional encodings to utilize efficient single-vector Maximum Inner Product Search solvers, which inherently supports incremental updates, making it naturally adaptable to streaming scenarios. 5. **Dessert** [5]: Compresses vector sets into sketches via Locality-Sensitive Hashing (LSH) [1] for fast approximate search.

To assess system stability in streaming scenarios, we compare StreamPLAID against two adaptations of PLAID[17]: **Frozen-PLAID**, which performs strict incremental updates with fixed centroids, and **ReindexPLAID**, which executes a complete re-clustering from scratch for each update.

5.1.4 Evaluation Metrics. Following recent retrieval literature [7, 8, 18], we evaluate ranking quality using Normalized Discounted

[1] At the inverted index level, Streaming Routing operates in a strictly append-only manner. Localized Repairing deregisters the posting lists of selected clusters and replaces them with newly re-clustered ones, leaving the rest of the index entirely unchanged.

[2] Updates sequentially traverse five LoTTE domains: Science → Lifestyle → Recreation → Technology → Writing. For evaluation, 10 queries are randomly sampled from the initial dataset, and 1 test query is subsequently drawn from the current active domain every 10 update steps.

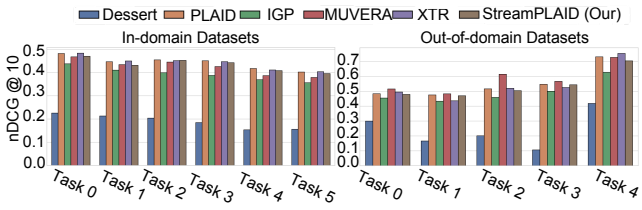


Figure 6: Incremental Effectiveness. Query effectiveness for incoming updates. The bars represent the average retrieval performance calculated across all evaluated datasets at each corresponding task for In-domain (left) and Out-of-domain (right) datasets.

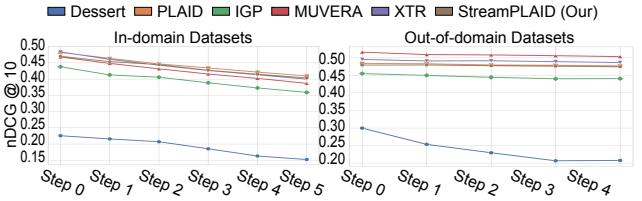


Figure 7: Long-term Effectiveness. Performance fluctuation tracked across continuous update steps for In-domain (left) and Out-of-domain (right) datasets.

Cumulative Gain at rank 10 (nDCG@10). Additionally, we assess index maintenance efficiency by measuring the indexing time per update batch. To ensure reliability, all experimental results except for the large-scale LoTTE-Pooled are reported as the average of three independent runs, unless otherwise specified.

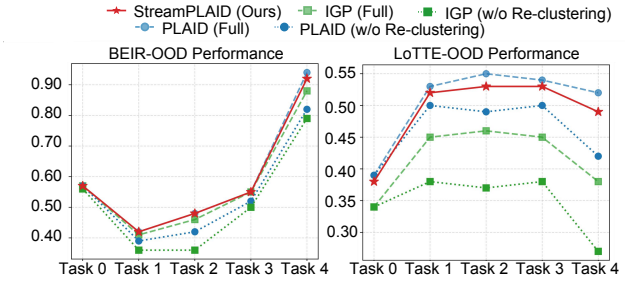


Figure 8: Performance on BEIR-OOD and LoTTE-OOD. StreamPLAID (*solid red*) tightly tracks the full re-clustering upper bound (*dashed*), successfully averting the severe concept drift degradation suffered by baselines without continuous updating (*dotted*).

5.2 Overall Comparison (RQ1)

5.2.1 ***Index Update Efficiency.*** Table 3 reports the average index update times. StreamPLAID ranks first across all seven benchmarks by a significant margin (e.g., only 512.28s on the massive LoTTE-Pooled dataset). By isolating updates strictly to degraded sub-clusters, **our topology-aware localized repair bypasses the prohibitive overhead of global reconstruction,**

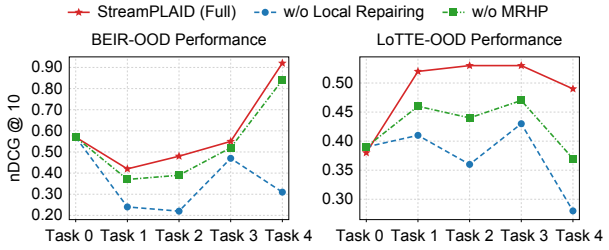


Figure 9: Ablation study on BEIR-OOD and LoTTE-OOD. StreamPLAID (*solid red*) effectively mitigates concept drift. Removing Local Repairing (*blue dashed*) triggers a catastrophic performance collapse, while omitting MRHP (*green dash-dotted*) consistently degrades accuracy, validating the necessity of topology-aware localized centroids.

resolving the write amplification problem and ensuring highly efficient temporal scaling.

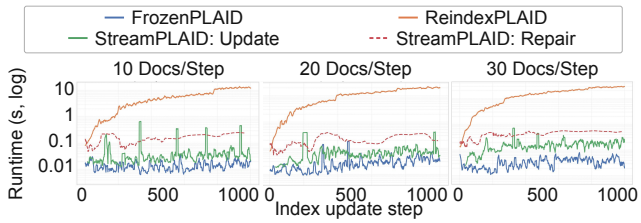
Conversely, baseline methods suffer from severe computational bottlenecks. **VQ-reliant architectures (PLAID [17], IGP [2], Dessert [5], and XTR [7]) require full global re-clustering** during updates to establish new centroids, resulting in exorbitant temporal costs. Furthermore, while MUVERA [4] supports incremental updates, **its mechanism of compressing multi-vectors incurs massive computational overhead, leading to excessively long update times.**

5.2.2 ***Index Update Effectiveness.*** To comprehensively evaluate index update effectiveness, we assess retrieval robustness from two distinct perspectives: query effectiveness on newly ingested datasets upon their first appearance (Figure 6), and the long-term performance trajectory of the initial task's queries (Task 0) across continuous update steps (Figure 7).

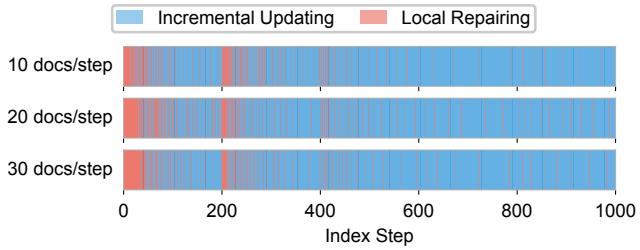
Overall, StreamPLAID consistently achieves top-tier accuracy across both in-domain and out-of-domain scenarios. For incoming updates (Figure 6), our method seamlessly adapts to novel data distributions, delivering retrieval precision that closely matches computationally expensive baselines like MUVERA [4] and XTR [7]. Furthermore, throughout the continuous streaming phase (Figure 7), StreamPLAID maintains a remarkably smooth and competitive performance trajectory. Unlike rigid index structures (e.g., Dessert [5] and IGP [2]) that suffer a precipitous decay in long-term effectiveness due to concept drift when centroids are statically fixed, our approach preserves structural consistency. Conversely, while XTR [7] and MUVERA [4] manage to maintain high accuracy over time, they achieve this solely at the expense of exorbitant computational overhead during updates. **Ultimately, StreamPLAID retrieves new documents effectively while preserving stability on historical queries.**

5.3 Impact of Re-clustering and Robustness of StreamPLAID (RQ2)

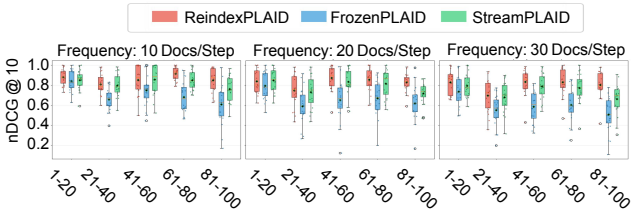
To evaluate our localized update mechanism, we compare StreamPLAID with PLAID and IGP across tasks T0–T4, using full re-clustering as an upper bound and fixed initial centroids as



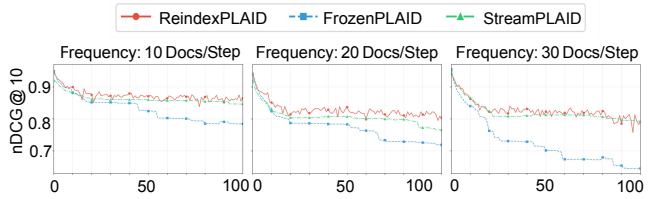
(a) Update Efficiency: Incremental index update time comparison (showing the first of five repeated experimental runs).



(b) Maintenance Pattern: Execution pattern per step of StreamPLAID, indicating whether an incremental update or local re-cluster was triggered.



(c) Incremental Effectiveness: Query effectiveness for incoming updates, visualized as box plots grouped every 20 tasks (averaged over 5 independent runs).



(d) Long-term Effectiveness: Performance fluctuation of the initial task's queries tracked across the entire continuous update phase (averaged over 5 independent runs).

Figure 10: Comprehensive evaluation under continuous data streams, detailing update efficiency, structural maintenance patterns, and retrieval effectiveness evolution.

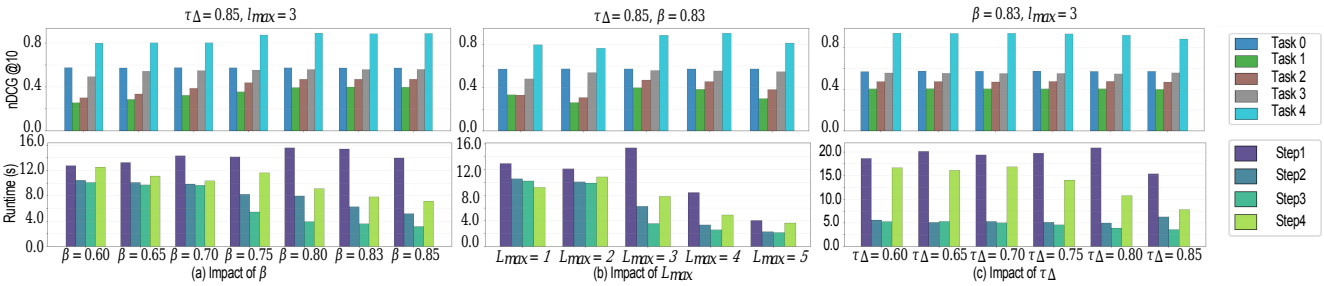


Figure 11: Hyperparameter analysis on different configurations. From left to right, the panels represent the impact of β (left), L_{max} (middle), and τ_Δ (right). For each parameter, the top figure shows the retrieval accuracy (nDCG) and the bottom figure presents the index update time. We evaluate the parameter sensitivity on the BEIR-OOD dataset. To ensure reliability, all reported results are averaged over three independent runs.

a baseline without re-clustering. As shown in Figure 8, StreamPLAID closely tracks the full re-clustering results on both BEIR-OOD and LoTTE-OOD, while the fixed-centroid baselines lose accuracy as the data distribution shifts. Together with the update-time results, these findings show that localized updates are needed to maintain retrieval quality without the cost of full re-clustering.

5.4 Ablation Studies (RQ3)

We evaluate two variants: (i) **w/o MRHP**, which collapses the hierarchy into a single layer but retains topology-aware subset extraction and localized repair, and (ii) **w/o Local Repairing**, which retains MRHP and relies only on Streaming Routing. Figure 9 shows that removing MRHP lowers retrieval accuracy on both datasets, with LoTTE-OOD nDCG@10 falling from 0.48 to 0.37 at T4, supporting MRHP's role in separating dense clusters from sparse outliers during repair. Without Local Repairing, BEIR-OOD nDCG@10 falls to about 0.31 at T4 as outliers accumulate in the bottom layer. This shows that Streaming Routing alone cannot maintain retrieval quality under concept drift.

5.5 Stability and Robustness of StreamPLAID under Continuous Ingestion (RQ4)

Figure 10 evaluates the maintenance efficiency and retrieval robustness of StreamPLAID under continuous ingestion. Overall, StreamPLAID is more efficient than ReindexPLAID and maintains higher long-term retrieval accuracy than FrozenPLAID.

Figure 10a shows that the maintenance time of StreamPLAID without local re-indexing (green line) is close to that of FrozenPLAID (blue line) across all three insertion frequencies. Local re-indexing (red dashed line) introduces some overhead but remains much faster than ReindexPLAID (orange line). As the document volume grows, StreamPLAID maintains stable update time, while the cost of ReindexPLAID increases sharply. This is achieved by the Topology-Aware Localized Repair mechanism described in Section 4.3. Figure 10b further shows when StreamPLAID performs local re-indexing and when it uses standard incremental updates.

Figure 10c compares retrieval performance on newly inserted documents. StreamPLAID achieves results close to ReindexPLAID, with similar median values and data distributions, while FrozenPLAID degrades as its static clusters fail to adapt to new data distributions. Figure 10d shows a similar trend for queries from the initial task. StreamPLAID remains close to ReindexPLAID throughout ingestion, whereas FrozenPLAID suffers a sharp decline due to its fixed cluster centroids. StreamPLAID also exhibits less performance fluctuation than ReindexPLAID, whose repeated global cluster reshuffling causes larger variations. These results show that localized maintenance helps preserve retrieval performance and structural consistency during continuous ingestion.

5.6 Hyperparameter Analysis

Figure 11 evaluates the effects of the retention ratio β, the number of layers $L_{\max}$, and the local dispersion threshold τ_Δ on retrieval accuracy (nDCG) and update time.

Retention ratio β. Accuracy degrades when β is small (e.g., 0.60) and stabilizes as it approaches 0.80. Larger β reduces the number of vectors reaching the final layer, lowering local repair overhead. We therefore use $\beta = 0.83$.

Number of layers $L_{\max}$. Accuracy improves with increasing $L_{\max}$, peaking at 3–4 layers before slightly declining at $L_{\max} = 5$, while update time decreases. Excessive depth may prevent drifted points from reaching the final layer for re-clustering, degrading accuracy. We use $L_{\max} = 3$.

Local dispersion threshold τ_Δ. Accuracy remains stable across the tested range (0.60–0.85), while update time decreases at $\tau_\Delta = 0.85$ by avoiding unnecessary local repairs. We therefore use $\tau_\Delta = 0.85$.

6 RELATED WORK

Multi-Vector Retrieval (MVR), pioneered by ColBERT [6], enables fine-grained semantic matching but poses challenges for dynamic indexing. Existing systems, including ColBERTv2 [18], PLAID [17], IGP [2], Dessert [5] with LSH [1], and XTR [7], rely on batch-oriented clustering or fixed partitions, requiring costly re-indexing under distribution shifts. In contrast, clustering-free approaches such as MUVERA [4] and CITADEL [8] support dynamic updates but face trade-offs between retrieval fidelity and storage efficiency.

Although streaming index maintenance has advanced in graph-based vector databases, including FreshDiskANN [20], IP-DiskANN [26], and topology-aware methods [28], these techniques cannot be directly transferred to MVR, whose MaxSim aggregation over token embeddings introduces distinct storage and computational challenges. Generative Retrieval [31, 33] explores an alternative retrieval paradigm, while hashing-based ANN indexes such as PDET-LSH [23] offer probabilistic accuracy guarantees and fast construction under a fixed corpus. Neither directly addresses efficient index-quality maintenance for continuously evolving MVR workloads, which motivates StreamPLAID.

7 LIMITATIONS AND FUTURE WORK

While StreamPLAID reduces index maintenance overhead while preserving retrieval accuracy, two limitations remain. First, the fixed retention ratio β may send semantic edge points to deeper layers, degrading candidate filtering. Second, fixed layer-wise radius thresholds and frozen intermediate centroids limit adaptation to local density variations, potentially accumulating quantization errors. Future work will investigate adaptive edge-point handling and dynamic centroid updates to further improve retrieval accuracy and maintenance efficiency.

8 CONCLUSION

We propose StreamPLAID, a dynamic indexing framework for Multi-Vector Retrieval (MVR) that integrates Monotonic Radius Hierarchical Partitioning (MRHP), Streaming Routing, and topology-aware localized repair to support continuous data ingestion without exhaustive global index reconstruction. Crucially, under bounded localized repair, this structural design reduces the per-update index maintenance complexity from a superlinear $\mathcal{O}(N\sqrt{N})$ cost to $\mathcal{O}(1)$ with respect to the continuously growing global corpus size. Extensive experiments across in-domain and out-of-domain datasets demonstrate substantial improvements in update efficiency while preserving competitive retrieval accuracy, and stability analyses validate its effectiveness under continuous ingestion. Looking forward, we envision StreamPLAID serving as the underlying index component of RAG engines in next-generation native distributed databases such as OceanBase [27].

ACKNOWLEDGMENTS

This work was supported by the Ant Group Research Fund. Lei Chen's work is partially supported by the National Key Research and Development Program of China (Grant No. 2023YFF0725100), the National Science Foundation of China (No. U22B2060), the Guangdong–Hong Kong Technology Innovation Joint Funding Scheme (Project No. 2024A0505040012), the Hong Kong RGC GRF Project 16213620, the RIF Project R6020-19, the AOE Project AoE/E-603/18, the Theme-based Project TRS T41-603/20R, the CRF Project C2004-21G, the Key Areas Special Project of Guangdong Provincial Universities 2024ZDZX1006, the Guangdong Province Science and Technology Plan Project 2023A0505030011, the Guangzhou Municipality Big Data Intelligence Key Lab (2023A03J0012), the Hong Kong ITC ITF grants MHX/078/21 and PRP/004/22FX, the Hong Kong ITC TC-SKLCRCC26EG01, the Zhujiang Scholar Program 2021JC02X170, the Microsoft Research Asia Collaborative Research Grant, the HKUST–WeBank Joint Research Lab, the 2025 HKUST Shenzhen–Hong Kong Collaborative Innovation Institute Green Sustainability Special Fund from Shui On Xintiandi and the InnoSpace GBA, and the Research on HKUST(GZ) Practices (ROP) Project (HKUST(GZ)-ROP2025001).

REFERENCES

[1] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya P. Razenshteyn, and Ludwig Schmidt. 2015. Practical and Optimal LSH for Angular Distance. In *Neural Information Processing Systems*. 1225–1233. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:9222460

[2] Zheng Bian, Man Lung Yiu, and Bo Tang. 2025. IGP: Efficient Multi-Vector Retrieval via Proximity Graph Index. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2524–2533. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:280071224

[3] Lianghao Deng, Yuchong Sun, Shizhe Chen, Ning Yang, Yunfeng Wang, and Ruihua Song. 2025. Muka: Multimodal knowledge augmented visual information-seeking. In *Proceedings of the 31st International Conference on Computational Linguistics*. 9675–9686.

[4] Laxman Dhulipala, Majid Hadian, Rajesh Jayaram, Jason Lee, and Vahab S. Mirrokni. 2024. MUVERA: Multi-Vector Retrieval via Fixed Dimensional Encoding. In *Advances in Neural Information Processing Systems 37*, Vol. 37. 101042–101073. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:270123558

[5] Joshua Engels, Benjamin Coleman, Vihan Lakshman, and Anshumali Shrivastava. 2023. DESSERT: an efficient algorithm for vector set search with vector set queries. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*, Vol. 36. Curran Associates Inc., Red Hook, NY, USA, Article 2974, 21 pages.

[6] O. Khattab and Matei A. Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 39–48. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:216553223

[7] Jinhyuk Lee, Zhuyun Dai, Sai Meher Karthik Duddu, Tao Lei, Iftekhar Naim, Ming-Wei Chang, and Vincent Zhao. 2023. Rethinking the Role of Token Retrieval in Multi-Vector Retrieval. In *Advances in Neural Information Processing Systems 36*, Vol. 36. 15384–15405. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:257921404

[8] Minghan Li, Sheng-Chieh Lin, Barlas Oğuz, Asish Ghoshal, Jimmy J. Lin, Yashar Mehdad, Wen tau Yih, and Xilun Chen. 2023. CITADEL: Conditional Token Interaction via Dynamic Lexical Routing for Efficient and Effective Multi-Vector Retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 11891–11907. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:253708231

[9] Rui Li, Quanyu Dai, Zeyu Zhang, Xu Chen, Zhenhua Dong, and Ji-Rong Wen. 2025. KnowTrace: Bootstrapping Iterative Retrieval-Augmented Generation with Structured Knowledge Tracing. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*. 1470–1480. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:278911303

[10] Weizhe Lin, Jinghong Chen, Jingbiao Mei, Alexandru Coca, and Bill Byrne. 2023. Fine-grained late-interaction multi-modal retrieval for retrieval augmented visual question answering. *Advances in Neural Information Processing Systems* 36 (2023), 22820–22840.

[11] Weizhe Lin, Jingbiao Mei, Jinghong Chen, and Bill Byrne. 2024. PreFLMR: Scaling up fine-grained late-interaction multi-modal retrievers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 5294–5316.

[12] Stuart P. Lloyd. 1982. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* 28 (1982), 129–136. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:10833328

[13] Sean MacAvaney and Nicola Tonellotto. 2024. A reproducibility study of plaid. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1411–1419.

[14] J. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, Vol. 1. University of California Press, 281–297. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:6278891

[15] Ramgopal R. Mettu and Greg Plaxton. 2002. Optimal Time Bounds for Approximate Clustering. *Machine Learning* 56 (2002), 35–60. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:1364387

[16] Jason Mohoney, Anil Pacaci, Shihabur Rahman Chowdhury, Umar Farooq Minhas, Jeffery Pound, Cedric Renggli, Nima Reyhani, Ihab F. Ilyas, Theodoros Rekatsinas, and Shivaram Venkataraman. 2024. Incremental IVF Index Maintenance for Streaming Vector Search. *ArXiv* abs/2411.00970 (2024). Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:273811165

[17] Keshav Santhanam, O. Khattab, Christopher Potts, and Matei A. Zaharia. 2022. PLAID: An Efficient Engine for Late Interaction Retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 1747–1756. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:248887228

[18] Keshav Santhanam, O. Khattab, Jon Saad-Falcon, Christopher Potts, and Matei A. Zaharia. 2022. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Vol. 1. 3715–3734. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:244799249

[19] D. Sculley. 2010. Web-scale k-means clustering. In *The Web Conference*. 1177–1178. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:6634147

[20] Aditi Singh, Suhas Jayaram Subramanya, Ravishankar Krishnaswamy, and Harsha Vardhan Simhadri. 2021. FreshDiskANN: A Fast and Accurate Graph-Based ANN Index for Streaming Similarity Search. *ArXiv* abs/2105.09613 (2021). Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:234790132

[21] Nandan Thakur, Nils Reimers, Andreas Ruckl'e, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1*, Vol. 1. 359–374. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:233296016

[22] Shijie Wang, Wenqi Fan, Yue Feng, Xinyu Ma, Shuaiqiang Wang, and Dawei Yin. 2025. Knowledge Graph Retrieval-Augmented Generation for LLM-based Recommendation. In *Annual Meeting of the Association for Computational Linguistics*, Vol. 1. 27152–27168. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:275336480

[23] Jiuqi Wei, Xiaodong Lee, Botao Peng, Quanqing Xu, Chuanhui Yang, and Themis Palpanas. 2026. PDET-LSH: Scalable In-Memory Indexing for High-Dimensional Approximate Nearest Neighbor Search With Quality Guarantees. *IEEE Transactions on Knowledge & Data Engineering* 38, 6 (June 2026), 3818–3834. https://doi.org/10.1109/TKDE.2026.3677369

[24] Jiuqi Wei, Zhenyu Liao, Ruoyu Han, Quanqing Xu, Chuanhui Yang, and Themis Palpanas. 2026. Taco: Data-adaptive and Query-aware Subspace Collision for High-Dimensional Approximate Nearest Neighbor Search. *Proceedings of the ACM on Management of Data* 4, 3 (2026), 1–28.

[25] Jiuqi Wei, Quanqing Xu, and Chuanhui Yang. 2026. The Virtuous Cycle: AI-Powered Vector Search and Vector Search-Augmented AI. In *2026 IEEE 42nd International Conference on Data Engineering (ICDE)*. IEEE Computer Society, Los Alamitos, CA, USA, 4120–4126. https://doi.org/10.1109/ICDE65706.2026.00309

[26] Haike Xu, Magdalen Dobson Manohar, Philip A. Bernstein, Badrish Chandramouli, Richard Wen, and Harsha Vardhan Simhadri. 2025. In-Place Updates of a Graph Index for Streaming Approximate Nearest Neighbor Search. *ArXiv* abs/2502.13826 (2025). Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:276449553

[27] Quanqing Xu, Chuanhui Yang, Feizhi Cai, Li Feng, Jiannan Ji, Zhifeng Yang, Guoping Wang, and Yi Zhang. 2026. PowerRAG: Building Smarter and More Accurate RAG Systems with OceanBase's Multi-modal Retrieval. In *Companion of the International Conference on Management of Data*. 142–145.

[28] Song Yu, Shengyuan Lin, Shufeng Gong, Yongqing Xie, Ruicheng Liu, Yijie Zhou, Ji Sun, Yanfeng Zhang, Guoliang Li, and Ge Yu. 2025. A Topology-Aware Localized Update Strategy for Graph-Based ANN Index. *Proceedings of the VLDB Endowment* 19, 3 (2025), 495–508. https://doi.org/10.14778/3778092.3778108

[29] Chia yuan Chang, Zhimeng Jiang, Vineeth Rakesh, Menghai Pan, Chin-Chia Michael Yeh, Guanchu Wang, Mingzhi Hu, Zhichao Xu, Yan Zheng, Mahashweta Das, and Na Zou. 2025. MAIN-RAG: Multi-Agent Filtering Retrieval-Augmented Generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 2607–2622. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:275212408

[30] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. BIRCH: an efficient data clustering method for very large databases. In *ACM SIGMOD Record*, Vol. 25. 103–114. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:1343506

[31] Zhen Zhang, Xinyu Ma, Weiwei Sun, Pengjie Ren, Zhumin Chen, Shuaiqiang Wang, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. 2025. Replication and Exploration of Generative Retrieval over Dynamic Corpora. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 3325–3334. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:278032939

[32] Yuxuan Zong and Benjamin Piwowarski. 2025. Towards Lossless Token Pruning in Late-Interaction Retrieval Models. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2407–2417.

[33] Ao Zou, Wenning Hao, Dawei Jin, Shichen Zou, Yu Zheng, Feiyan Sun, and Li Xiang. 2024. An Efficient Corpus Indexer for dynamic corpora retrieval. *Expert Syst. Appl.* 254 (2024), 124306. Retrieved October 28, 2026 from https://api.semanticscholar.org/CorpusID:270130274