ADB Report

Abdallah Ahmed Mohamed	1220251
Seifeldin Sameh Dakroury	1220246
ElHussein Awad	1220226

1. Introduction

After evaluating multiple indexing strategies for large-scale vector search, our team adopted an IVF-PQ. This approach offers an effective balance between accuracy, memory efficiency, and retrieval speed. Unlike IVF-Flat, which stores full vectors in memory-mapped form and can cause excessive RAM usage, IVF-PQ stores compact PQ codes, enabling scalable search on tens of millions of 64-dimensional vectors while keeping memory consumption low.

---

2. System Structure

Our system is composed of three major components:

1. Binary file storing all original vectors

   The full vectors remain stored in a fixed-layout binary file, but they are not loaded during normal retrieval. This preserves the ability to re-train PQ codebooks or rebuild the index without inflating RAM usage.

2. Disk-based IVF-PQ index

   The index contains:
   - k-means centroids
   - PQ codebooks
   - compressed PQ codes stored inside inverted lists, along with vector IDs

   Because PQ codes are compact (a few bytes per vector), the entire index remains small, efficient, and disk-friendly.

3. Retrieval layer with batched PQ decoding

   During querying, only PQ codes from selected inverted lists are loaded and processed in small batches, preventing system memory from being overwhelmed even when lists contain millions of codes.

The components work together to support efficient retrieval while minimizing both CPU load and RAM usage.

---

3. Index Construction

Sampling Strategy

Both centroid training and PQ codebook training require samples of the data. We draw large representative samples that capture the distribution without requiring full-dataset loading. This ensures high-quality clusters and PQ codebooks while keeping training manageable.

Training Centroids and PQ Codebooks

- Mini-batch k-means is used to compute a few thousand centroids.
- Product Quantization codebooks trained using Mini-batch k-means as well, we experimented with different PQ configurations to balance memory usage, accuracy, and decoding cost.

Assigning Vectors to Clusters and Compressing Them

For each vector:

1. It is accessed directly from the binary file.
2. Its nearest centroid is computed.
3. It is encoded using the PQ codebooks.
4. The resulting PQ code, along with its ID, is appended to the inverted list on disk

This structure avoids storing full vectors in the index and significantly reduces memory requirements.

Writing the Index

All index components (centroids, PQ codebooks, inverted lists) are written into a dedicated index directory. Their compact size reduces disk footprint and accelerates disk I/O during retrieval.

---

4. Retrieval Method

Query Preprocessing

A query is normalized and compared with all centroids. The centroid file is small enough that loading it incurs negligible overhead.

Selecting Clusters

Distances to all centroids are computed, and an nprobe is selected for search. Larger nprobe values increase recall at the cost of more candidate vectors.

Gathering and Decoding Candidates

For each selected cluster:

1. The inverted list file is opened.
2. PQ codes inside the list are loaded in small batches.
3. Each batch is decoded and compared with the query.

This batching mechanism is essential for scalability, it prevents excessive RAM usage even when clusters contain millions of vectors.

Ranking and Returning Results

The top candidate IDs from the heap are extracted. Full vectors are retrieved for these candidates. Cosine similarity is computed against the normalized query to refine rankings. The final top-k vector IDs are returned in descending order of similarity.

---

5. Design Trade-Offs and Parameter Decisions

Number of clusters

More clusters means bigger nprobe and shorter lists.
Fewer clusters result in larger lists and smaller nprobe.
Optimal choice would be the rule of thumb which is square root of db size.
We choose k to balance these effects for tens-of-millions scale.

PQ parameters

Higher compression reduces memory but hurts recall.
We evaluated multiple combinations to find the best balance.

nprobe value

Probing more clusters improves accuracy but increases candidate load.
Our experiments tune nprobe for optimal recall-latency balance.

Batch size during decoding

Larger batches improve CPU efficiency but increase memory usage.
We tuned this parameter to avoid RAM pressure.

Sampling size for training

Larger samples improve centroid and PQ quality but cost more computation.

---

8. Conclusion

Our updated system uses IVF-PQ to support large-scale vector retrieval efficiently and reliably. By combining k-means clustering with PQ compression and batched decoding, the system maintains low memory usage, fast query times, and strong retrieval accuracy. The architecture is robust, storage-efficient, and scalable to tens of millions of vectors, making it well-suited for our operational constraints.