Response to Reviewer Comments: " An Experimental Evaluation of Hybrid Querying on Vectors (EA&B)" Submission to VLDB 2026 Research Track (ID)

Jiaxu Zhu, Jiayu Yuan, Kaiwen Yang, Xiaobao Chen, Shihuan Yu,
Hongchang Lv, Yan Li, Bolong Zheng

Dear meta-reviewer and reviewers:

Thank you for giving us the opportunity to revise our paper. We are deeply grateful for the reviewers' constructive feedback and the comprehensive suggestions provided in the meta-review. We believe that we have addressed all the major issues and concerns, and the changes are highlighted in blue according to the reviewers' comments.

Response To Meta-reviewer

Summary Comments. *The paper empirically studies performance of hybrid querying methods with respect to two types of queries: attribute filtering (AF) NN search, range filtering (RF) NN search. This is a timely problem with high practical relevance. Nevertheless, several improvements would benefit the paper. We therefore recommend a revision. The main revision points are summarized below, however, we expect that the authors do their best in addressing all points raised in the individual reviews.*

Response:

R1. *Improve the readability of some figures (e.g., performance under different attribute distributions or selectivity).*

Response:

R2. *Run experiments with multi-modal queries (e.g., combining text and images) or more complex query types involving both attribute and range filters.*

Response:

R3. *Instead of using the point ID, run experiments with meaningful attributes for RF-NN search on datasets (Deep, YT-Audio, WIT).*

Response:

R4. *Evaluate if the server setting affects the relative ordering of methods in experiments*

Response:

R5. *In Table 1, please clarify which methods are disk-based, and which methods are memory-based. For disk-based methods, it is also meaningful to report the cost breakdown (e.g., in terms of I/O time and CPU time).*

Response:

R6. *Report the peak memory usage of query processing.*

Response:

R7. *Run Scalability experiments with larger datasets.*

Response:

R8. *Clearer insights and guidelines wrt the strength and weaknesses of the compared system.*

Response:

R9. *Clearer guidelines in the Discussion to the extend of recommendations on which system/algo to use for what dataset characteristics.*

Response:

Response to Review 1

W1. *Although the authors justify the synthetic attribute generation, real-world applications may involve more complex attribute semantics and correlations, which might affect generalizability.*

Response:

W2. *Some figures (e.g., performance under different attribute distributions or selectivity) are dense and could benefit from cleaner presentation or summarization to improve readability.*

Response:

W3. *It would strengthen the paper to delve into multi-modal queries (e.g., combining text and images) or more complex query types involving both attribute and range filters.*

Response: We think this is an excellent suggestion. We have enhanced our experiments by incorporating more complex multi-modal dataset evaluations in Section XXX of the paper. Initially, our paper utilized the Text2Image dataset, which is commonly used in existing multimodal ANN algorithms. This dataset presents a modality gap, as the base vectors are images while the query set consists of text, often requiring algorithms to exhibit stronger robustness for good performance. To further enrich the evaluation

of complex multimodal queries, we selected 500,000 text vectors and 500,000 image vectors from the Text2Image dataset to form a mixed base query dataset. Similarly, we chose 5,000 image vectors and 5,000 text vectors as query vectors, combining both text and image modalities. We then performed attribute filtering and range queries on this multimodal dataset to further analyze algorithm robustness. The relevant experimental results are presented and discussed in SectionXXXof our paper.

Response to Review 2

W1. *In the experimental study, datasets did not originate from the real applications for hybrid queries. Regarding the datasets for AF-NN search (Msong, Audio, SIFT1M, GIST1M, GloVe, Enron), the attributes were generated synthetically but not real. Regarding the datasets for RF-NN search (Deep, YT-Audio, WIT), the data point ID was used as attribute; however, it is not meaningful to issue range query on data point ID. In order to match with the introduction, the authors are recommended to include a real dataset from e-commerce platform with both vectors and attributes.*

Response:

W2. *All experiments were conducted on the same server mentioned in Section 4.1.3. However, in practice, server providers may use other hardware settings (with other processors and RAM). Would the server setting affect the relative ordering of methods in experiments? For example, if Intel Xeon processor is used, would there be any change of the top 3 methods in Figures 4,5,7,8? Would the CPU cache size affect the top 3 methods in those figures?*

Response:

W3. *In the experimental setup, it is unclear whether data storage (disk or SSD) is used. In Table 1, please clarify which methods are disk-based, and which methods are memory-based. For disk-based methods, it is also meaningful to report the cost breakdown (e.g., in terms of I/O time and CPU time).*

Response: Thank you for your suggestion! Currently, most hybrid query methods are based on memory, except for filteredDiskANN, which supports SSD, so we will add explanation information in the paper. At the same time, we also use real data sets to compare the difference in search performance between filteredDiskANN in memory and on SSD. The results are shown in Figure 1:

In response to your request for a breakdown of I/O and CPU cost in disk-based methods, we conducted additional experiments on the real-world yt8m dataset. We compared three variants of DiskANN:

- SSD-based FilteredDiskANN (original disk-based method)
- FilteredVamana (memory-based method)
- StitchedVamana (memory-based method)

All experiments were run with 32 search threads for fair comparison. The following figure shows the mean latency (log-scale, left y-axis) and I/O vs CPU proportion (right y-axis) across different Recall10 levels:

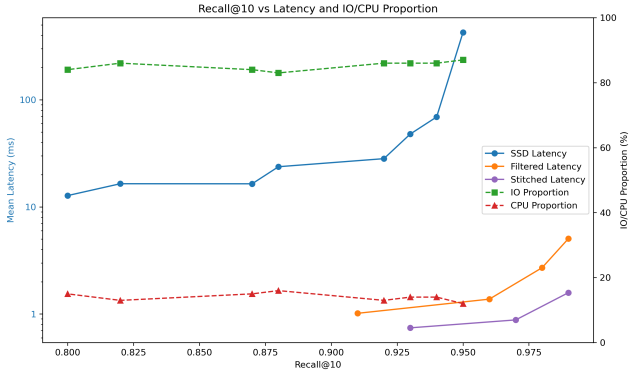

Figure 1

Key Observations:

Latency-Recall Tradeoff: As expected, SSD-based FilteredDiskANN exhibits significantly higher latency than in-memory methods, especially at higher recall levels. When Recall@10 reaches 0.96, its latency increases super-linearly to over 100 ms.

Cost Breakdown of DiskANN:The I/O proportion consistently dominates, maintaining 80%–85% of the total latency even at different recall levels. The CPU proportion remains low (10–15%), indicating that I/O is the major bottleneck in disk-based search pipelines. This clearly shows that improving disk read efficiency or reducing I/O volume is critical for performance optimization.

In-memory Methods:

FilteredVamana achieves a good balance, staying under 30 ms while achieving Recall@10 > 0.97. While StitchedVamana achieves extremely low latency (sub-5 ms) due to its stitched multi-label index structure, its fixed edge connectivity may limit effectiveness under highly uncorrelated label distributions. Nevertheless, it consistently supports 90%+ Recall@10 across diverse datasets, as shown in our evaluations.

W4. *The authors have reported the peak memory usage of index construction only. They are also recommended to report the peak memory usage of query processing.*

Response:

D1. *The authors have cited an experimental paper [49], but have not discussed about the main difference between their paper and [49].*

Response: Thank you for your suggestion. The paper [49] is a comprehensive survey and evaluation of graph structure approximate nearest neighbor search algorithms. Our paper extends this to a more complex "hybrid query" scenario and evaluates approximate search algorithms that combine vector similarity and attribute filtering/range filtering. Since hybrid queries are developed based on ANNs, we refer to the metrics and evaluation architectures of different methods proposed in this paper.

D2. *In Section 1.1, I could not find convincing references for the requirements in practical applications. [12] and [36] are research papers only, but not from real applications.*

Response:

D3. *In Definition 2.3, s_1 and s_n should be replaced with a_1 and a_n , respectively.*

Response: We sincerely apologize for the misuse of notation in Definition 2.3. This has been corrected in the revised version. Thank you for pointing it out.

D4. *In Section 4.1, please clarify the URL of the implementation of each method.*

Response: Thank you for the suggestion! Since there are multiple methods, listing all URLs in Section 4.1 would be overwhelming. Instead, we have added the corresponding URLs in footnotes on the relevant pages throughout the paper.

Response to Review 3

W1. *Scalability experiments are missing; largest dataset is 10M; some pitfalls will only show at large scale.*

Response:

W2. *The explanations provided with the performance results are not as insightful as they need to be to point out the pros and cons of the different algorithms.*

Response:

W3. *I'd expect general, clearer guidelines in the Discussion to the extend of recommendations on which system/algo to use for what dataset characteristics. Hence, the comprehensive perspective on their strengths and weaknesses of the approaches is missing.*

Response:

D1. *Section 2.2. Graph based and IVF are well known. Missing in the index description is the implementation of attributes for AF and RF as the section is call hybrid ANN.*

Response: Thank you for your suggestion! We recognize that the original manuscript did not clearly explain how attribute filtering (AF) and range filtering (RF) are embedded into standard Graph-based and IVF-based index structures.

To address this, we have revised Section 2.2 to summarize how the algorithms discussed in this paper implement hybrid query functionality based on these foundational indexing structures (with detailed implementation provided in Section 3):

For Graph-based indexes, we describe the typical approaches of embedding attribute and range constraints by associating metadata with graph vertices or edges. Specifically, attribute filtering (AF) is usually achieved by annotating vertices or edges with attribute information, enabling the traversal algorithm to skip nodes or edges that do not satisfy the query predicates (e.g., NHQ, Filtered-DiskANN, ACORN). In contrast, range filtering (RF) typically involves annotating edges with valid attribute intervals so that traversal proceeds only along edges that satisfy the query's range constraints, effectively reducing the search space (e.g., SeRF, DSG).

For IVF-based indexes, we point out that their inherent pre-filtering mechanism naturally supports hybrid query functionality. In particular, attribute and range filtering can be implemented by directly excluding irrelevant vectors during intra-cluster distance computations. Furthermore, we highlight the hierarchical partitioning approach based on attributes, which further divides clusters into finer-grained sub-partitions, significantly improving filtering efficiency (e.g., CAPS, PUCK).

D2. *Section 4.2.1: It is important to run the experiments on larger datasets, e.g. 100 M vectors, or 1B bigann. Otherwise, differences will not show. Also, build time parameters such as Radius and Length which impact the Recall need to be considered. Also, approaches for partitioning and merging for building very large indexes need to be considered. Please also document whether quantization is included in index build time. There is no explanation why Vamana is not included in Figure 7.*

Response: Thank you for your insightful comments!

Regarding dataset scale and scalability:

We have elaborated on the scalability implementations of various algorithms on the 100-million-scale Deep dataset in Section XXX of our paper. We also provide a comprehensive analysis of their performance on large-scale datasets.

Regarding the experimental parameter settings:

During index construction, for datasets where recommended parameters were explicitly provided in the original papers of the respective algorithms, we strictly adhered to those settings.. For the remaining datasets, we used the default parameters provided in their open-source code repositories. All detailed configurations of experimental parameters are publicly available in our code repository.

Our experimental results show that on datasets where recommended parameters were used, the relative performance ranking of the algorithms did not change significantly. During the search phase, we obtained performance data at different Recall and QPS by adjusting the search parameters. These search parameters covered a wide range to ensure we could obtain comprehensive experimental result curves.

Given the large number of datasets and algorithms involved, to maintain the conciseness of the paper, we did not list the detailed running parameters for all algorithms and datasets in the main text. These specific parameters can all be found in our code repository. For algorithms that are sensitive to specific parameters, we have highlighted them in the relevant sections of the paper.

Regarding index partitioning and merging:

For partitioning and merging, only Filtered-DiskANN, which supports disk storage, builds indices separately and then merges them. However, the algorithms we focused on and evaluated in this study (with the exception of Filtered-DiskANN) are all designed

and implemented based on in-memory index construction. Therefore, for the vast majority of algorithms, index partitioning and merging are not involved, which differs from DiskANN's design considerations.

Regarding quantization time in index build:

f all the evaluated algorithms, only Faiss supports quantization. However, in our experiments, we used the IVF index structure of Faiss without employing quantization. Therefore, no quantization time was included in the reported index build times.

Regarding Figure 7 and the exclusion of Vamana:

We appreciate the reviewer's observation. Figure 7 presents experiments on multi-label construction and search, using Boolean AND logic. The Vamana-based methods (FilteredVamana and Stitched-Vamana) only support Boolean OR logic for multi-label search, which is why they were not included. We will clarify this point in the revised version of the paper.

All the evaluated algorithms in our experiments build the index in a one-shot manner, without applying any partitioning or merging strategies. Additionally, although some of the methods we used—such as FAISS, PUCK, DiskANN, and Milvus—support vector quantization during index construction, we did not enable quantization in any of our experiments. Therefore, the reported index build times do not include quantization.

D3. *The paper lists AF algos (table 1) and RF algos (table 2). It would be good to list the systems in a table with their algos and capabilities (AF, RF, Graph, IVF, ...). Would make it easier to follow the experimental comparison. E.g., Fig 12 does not include Vamana/DiskANN. Should I assume that RF is not possible with Vamana, or just not implemented?*

Response:

D4. *Section 4.3. It is not explained, what the Range attributes are, or how many attributes. I assume 1 attribute. The index construction times are vastly different in Fig 12 a. This calls for a much more detailed explanation. Also, Milvus seems to be missing in Fig 12 c. Why?*

Response: Thank you very much for your careful review and valuable suggestions. We greatly appreciate your feedback and sincerely apologize for any lack of clarity in our paper.

Regarding the issue of range attributes, we indeed used only one range attribute for evaluation. At present, most existing range filtering algorithms only support filtering on a single attribute. This is a major limitation of current range filtering methods. We have clearly pointed this out in Section 5.2.2 of the "Discussion" part of the paper and listed it as an important challenge for future work. Thank you for highlighting this point, which further emphasizes the significance of our work.

In response to the large difference in index construction time shown in Figure 12a, we have revised the paper based on your suggestion. We added more detailed and in-depth analysis to provide a clearer explanation.

Finally, your observation about the missing peak memory usage data for Milvus in Figure 12c is very insightful. Initially, we hesitated to include the peak memory usage of databases (including Milvus) because they are full systems that run continuously and contain many components. This makes them quite different from other algorithms.

However, we agree with your point. To provide a more comprehensive evaluation, we have added the peak memory usage data of the database systems (including Milvus) in the revised version of the paper. This offers a more complete comparative view.

Once again, thank you for your constructive comments. Your valuable feedback has helped us improve the paper further.

D5. *Section 4.3.2: This is a very vague explanation. "SeRF exhibits relatively weak overall performance, with a significant drop in recall under small-range query scenarios. This suggests that while its compressed graph structure is space-efficient, it has a substantial negative impact on query performance." It would be important to be specific about what in the design/architecture of the SeRF graph is causing this behavior.*

Response: Thank you for your feedback. We have revised Section 4.3.2 in our paper to provide a more specific and clearer explanation.

Specifically, SeRF compresses the entire HNSW graph collection by adding range information to the edges. While this approach saves space, it leads to significant performance degradation when executing small-range queries. This is because SeRF's multi-filtering mechanism needs to inspect a large number of edge entries that are irrelevant to the current small-range query, thereby introducing fixed computational and memory access overhead. Additionally, during graph traversal, many invalid neighbors may be visited, causing the search to fall into local optima, which further degrades query performance and recall.

We believe this explanation better clarifies the specific reasons why SeRF performs poorly in small-range query scenarios. Thank you again for your valuable suggestions.