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:

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:

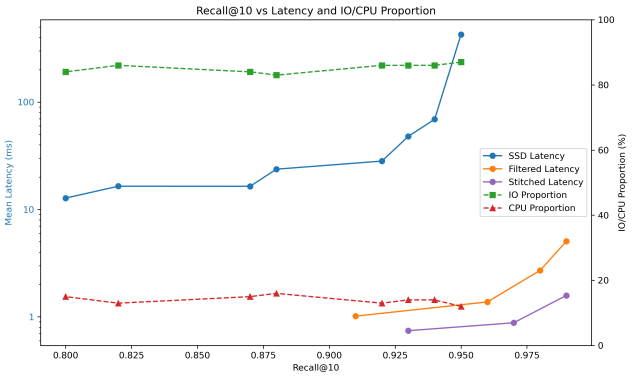

Figure 1

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:

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:

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

Response:

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:

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: Thanks to the reviewer for suggesting that there is no explanation in Figure 7 why Vamana is not included. Figure 7 is an experiment on multi-label construction and search, and the Boolean logic of multi-label search is AND. The two algorithms using Vamana, FilteredVamana and StitchedVamana, only support Boolean logic OR for multi-label search, so we did not include them. We will add an explanation of this in the paper.

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:

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: