Computer Fraud and Security

ISSN (online): 1873-7056

Vector Search Optimization for Scalable Information Retrieval in Low-Latency Applications

Anuraag Mangari Neburi

Vice President

ABSTRACT: In this paper, it is analysed how the modern search systems of vectors are made faster, more accurate and can be easily scaled. It is based on qualitative method in order to review the ideas of research articles, a system within the industry and technical reports. According to the results, it may be reduced with the help of good index structure, adaptive search algorithms and strong design of hardware and software that minimizes the latency and improves the recall. The memory tiering, graph indexes and hybrid search models and their use in supporting large workload e.g. RAG and recommendation are also outlined in the paper. The results show that small contribution of algorithms, memory and hardware in a large number are added to make sure that in real life, the search of vectors is fast and efficient.

KEYWORDS: Latency, Optimization, Scalability, Vetor

I. INTRODUCTION

Since chatbots, fraud detection, and recommendation engines, among others, are now relevant AI systems, many of them include vector search as an essential component. Such systems have to locate the nearest vectors in a short amount of time, with millions or even billions of items. Due to this necessity, numerous new techniques were devised to render the process of a vector search fast and scalable. In the current paper, the methods are presented and the way they operate in actual systems will be described. It is concerned with index layouts, adaptive algorithms, memory layouts and hardware selections. This paper aims at providing an easy and clear explanation of how the modern-day vector search can be made efficient.

II. RELATED WORKS

Approximate Nearest Neighbor (ANN) Algorithms

Scalable search The ANN search has made itself the target of scalable vector search procedures since it allows the system to trade-off stated accuracy to gain significantly quicker service. ANN algorithms classical are rooted on compression of vectors, index pruning and selective search expansion to further cut the expense of calculations in large datasets. The trade-off between time and accuracy is one of the biggest problems of ANN especially when systems are attempting to support the remaining percentages of recall.

The majority of the existing ANN algorithms possess some predetermined stopping criteria, or put another way, perform the same amount of exploration by each search move regardless of the difficulty of the query. It is a set strategy which leads to unnecessary search on simpler queries and under search on more complicated queries.

The stated limitation can be addressed using the adaptive early-termination method proposed in [1] that trains variable stopping points of every query using decision trees that are boosted by the gradient. The degree to which additional exploration is necessary to arrive at an answer is as satisfactorily computed by retrieving runtime indications of the intermediate search states.

This technique permits a reduction in the latency to as high as 7.1 times across the same accuracy as ANN search with a fixed configuration. The argument here is that ANN difficulty varies not just among queries, but it can be seen that adaptive practices can remove computation wastage. This is important to low-latency applications where the response time is to be less than 50ms since adaptive termination reduces tail latency spikes.

The other major problem with ANN search is the support of the filtered retrieval as well. ANN search on subsets in real world systems might be required depending on metadata which can be timestamps, product category or language. The case of filtered retrieval is not suitable to the traditional ANN indices because it uses the geometric proximity and not the attribute constraints.

In [2], the article has been written in the format of a graph wherein the connections are joined together with labels. The method suggests that there will be a major enhancement in the latency of the filtered-query when the relationships of label closeness have been incorporated in the index along with the geometrical proximity. One of them could be made to stream construction and the other one could be made to index in batches.

They are a thousand times faster than conventional filtered ANN algorithms, and can be used to answer thousands of queries per second directly out of SSDs. This is critical to recommendation systems and search systems where filtering attributes to a certain degree is a distinguishing attribute.

ANN algorithms that are based on graphs continue to be a type of leading algorithms because they have high recall and convergence rate. The traditional proximity graphs take up memory and they are not viable as the size grows.

An appropriate option would be the one by the Routing-guided Product Quantization (RPQ) algorithm in [3] that unites Product Quantization with routing characteristics of proximity graphs. Compared to classical PQ that quantizes vectors without considering graph relationships between them, RPQ encodes routing structure during end-to-end differentiable training into quantized codes.

This gives tremendous improvements to the quality of the quantized vectors and the overall performance of graph searches. Experiments reveal that query-per-second throughput of RPQ is 4.2 times greater than at the same recall rate at 10, demonstrating how closely linked learning and searching a graph may open the effectiveness and performance of billion-scale applications.

Memory and Hardware Co-Design

Storing, memory design does not make the computation speed of the algorithm any less significant than the increase of a data as a vector to billions of elements. This is due to the fact that the first bottleneck is that indexes of embedded SSDs cannot be accessed in an appropriate way (vector search require must finely random read) but SSDS cannot provide good through put when accessed as a coarse sequential access.

This is one of the gaps which are considered in the work in [4] which proves that the indexing throughput of the SSD-based improvement could be regularly obtained at the price of the astronomical increment of the index-size: index-size could be increased up to the moment of 7.7 times of the dataset size. It makes SSD-only solutions highly costly and unproductive.

As the authors say, it is possible to form an idea of exploiting the second-tier memory remote DRAM or NVM that is connected to CXL or RDMA and could potentially provide optimum granularity to the processes of a vector index. This they can do by developing new graph and cluster index designs that can exploit the fact that this memory layer is low-latency, which can execute virtual optimum search performance using much lower storage overheads.

At the same time, a hybrid algorithm of search graph and quantization is being created. The acceleration technique proposed in the article [5] increases the speed of the implementation of the IVF-HNSW by adding adaptive termination and minimizing the number of the points that were accessed with the assistance of the HNSW-based screening.

It may be analogized to adaptive early-exit logic of [1] which is applied in the case of heavy quantization searches. The results of these findings provide significantly most of the gains in search speed due to the decrease of the operations of the probes per query. This is very helpful in the industry where the PQ clusters have been growing to colossal proportions and the traffic within the query is very high.

The other new direction of acceleration of the vector search has been codesigning of the software with hardware. The near-data processing (NDP) systems also augment the calculations around the stores with the aim of reducing the cost of transfer of information. The NDSEARCH architecture, of [7], extends the operators of the vector searches in the NAND flash itself that is the extension of the SEARSSD architecture.

This is what facilitates the parallelism in the LUN-level and reduced host computing. The NDSEARCH is 31.7 times faster than the CPU based designs, the GPU based designs and SmartSSD only design and has an enormous energy saving. Those are the ones that are pioneered by the trend of pipeline specialization of inference using special hardware in real-time.

The second hardware friendly development is Falcon accelerator and algorithm of Delayed-Synchronization Traversal (DST) [10]. Falcon is made simple to implement on-chip Bloom filters and memory efficient traversal logic. DST improves the utilization of the graph search in reversing the order of traversing the route to be more congruent with the activity of accelerators.

They are also fast and up to 19.5x speedy and highly efficient in energy. These contributions in hardware indicate that the hardware in the future of hardware-based systems of vector search will not be smarter algorithms, but on high-level hardware path designs of a vector search application.

End-to-End Optimization

ANN indexing and search strategies have also started going through deep learning methods. Research including [8] points to a new trend in which heuristics are being replaced by data-driven learnable indexing solutions. Deep learning is

able to learn patterns of distributions not fully exploited by traditional multi-step ANN pipelines which require high-dimensional embeddings.

These involve the acquisition of partition boundaries, quantization centres and even search paths right out of data. The survey identifies the methods as learning to index and learning to search as both of them help in making the retrieval pipelines more adaptive and efficient.

The indexing based on learning is specifically effective as it learns behavior based on the statistical characteristics of the embedding space. One such system is called RPQ [3] and the quantization is directed by routing patterns derived out of existing PG structures.

Adaptive termination of [1] is a technique that operates based on learned models and it minimizes redundant search operations using measured intermediate signals. The increasingly intricate datasets and embeddings (in particular, multimodal inputs) render the existing tactical indexing approaches inadequate.

Other examples of the deep-learning-powered future include [9] that suggests an unsupervised, patience-based early-exit methodology of A-kNN. The technique determines easy queries that their closest neighbors fall under a limited number of clusters and hard queries that one needs to search further.

The cascade mechanism is a combination of fast pre-checking and dynamic-depth scanning and results in up to 5x speedups with insignificant losses in effectiveness. This is consistent with the trend in general to adaptive ANN systems, in which depth, termination and routing are determined by the query attributes and not by a priori rules.

There are also new problems in retrieval with increased data modality. Other hybrid approaches such as ACORN [6] combine the similarity between vectors and flexible structured predicates. In contrast to previous hybrid systems, which can only support a limited set of query types, the predicate-subgraph traversal of ACORN can support a variety of predicate sets, and still, be able to achieve high performance.

The performance has been evaluated to be ACORN performing 2-1000 times greater throughput than the previous hybrid search schemes. Such flexible index structures are needed as the enterprises are moving towards multimodal retrieval (text, image, attribute and keyword constraints).

Multimodal Vector Search

The contemporary retrieval systems should be scalable to be applied in conversational AI, fraud detection, personalization and RAG applications. Graph systems are highly recollective and ineffective in relation to memory usage with the expansion of datasets.

Quantization-based algorithms are memory-sparing (precision or query-performing cost) algorithms. These trade-offs are attempted to be solved by adaptive mechanisms with hybrid and learned systems. The available information in all the studies [1] -[10] is suggestive of a multitude of common themes:

- There is the need to adjust to the computation. Techniques that are used at the early exit reduce the latency at a high level of recall loss.
- The routing must be routing conscious. RPQ shows that the knowledge of the topology of the graph improves the performance of PQ.
- It is highly dramatic that hardware and algorithm co-design are enhanced in terms of throughput. Falcon and NDSEARCH display the orders-of-magnitudes.
- Hybrid multimodal search is the way to go. The systems must be able to circulate systemized search as compared to disorganized search.
- Second-level memory will be used to replace SSD-heavy architectures. Storage storage is highly granular and therefore, affects the performance of vector searches.

Commercial systems are experiencing a constriction in the latency requirements, so in the future indexing strategies will more often combine learning-based optimization, multiple-tier memory system design, and computer-coded acceleration.

Computation depth dynamically controlled by search pipelines depending on the complexity of queries and a search involving vectors will be extended to multimodal and federated environments with reduced privacy and memory constraints. The existing sources all lead to one definite conclusion scalable vector search needs to integrate both algorithmic efficiency and adaptive intelligence with hardware-aware implementation.

III. METHODOLOGY

The qualitative research methodology chosen in this study is to learn how the modern vector search systems would be able to facilitate the low latency, high recall, and scalable performance, in the real-life settings. They are further supported by the fact that most design decisions, such as index structures, hardware boards, memory hierarchy and adaptive algorithms, influence the optimization of the vector search, hence a qualitative method can be employed to study these aspects in detail. Instead of putting the numerical performance to a test, the ambition will be to put the ideas, trends, trade-offs, and design strategies to a test as provided in the extant literature, industry systems and technical literature.

The paper begins with thematic searching of higher methods of vectoring searches. The selection of ten external literature sources is based on the fact that these resources explain the important breakthroughs in the framework of the approximate nearest neighbor (ANN) search, adaptive termination techniques, learned indexes, hybrid search, product quantization, graph-based indexing, second-tier memory designs, and acceleration of near-data processing.

The papers are first read so as to come up with shared themes in the shapes of search efficiency, memory reduction, adaptive algorithm, hardware-software co-design approach and multimodal or filtered search support. These themes form a systematic knowledge of the optimization of the search systems of low latency vectors.

A comparative analysis is carried out in comparing different optimization strategies. It includes comparison of fixed ANNs and adaptive ANNs, comparison of disk-based indexes and second layer memory architectures as well as simple methods of HNSW and IVF-PQ and more complex learned quantization, or hybrid search methods.

It is not geared towards quantifying some exact raw numbers but finding out which concepts will always be useful to reduce latency, the index size smaller or the recall high. The comparison is also helpful to reveal the common drawbacks such as, the memory of the graph-based indexes, the accuracy-latency trade-off when quantizing products, and the issue of storing vectors of billions of products in SSDs.

This is succeeded by a synthesis conceptualization generated through mapping of interaction of numerous elements of the vector search system. As an illustration, the algorithms, indexing structures, as well as the hardware accelerators are studied together and understand the impact of a modification in one of the layers on the other layer.

The knowledge of the opportunities of adaptive early exit algorithms to the unnecessary computation, the predicate-conscious graph-traversal optimization plans multimodal or filtered search, and the near-data processing to make the computation even closer to the storage is contained in this synthesis, to cut the I/O delay to a minimum. Other future trends of synthesis include the trending patterns such as end-to-end learned indexing, hybrid semantic-structured search, transformation of vector search into distributed, memory tiered systems.

The paper will proceed to examine how far the techniques can be beneficial in supporting low-latency applications such as RAG, fraud detection, personalization, and edge AI. This is done through the description of how each of the references that characterize the practical loads and what optimizations were needed to achieve the interactive response times. The acquired qualitative data is used to correlate the academic innovations with real needs in the sphere of engineering which entails high concurrency, predictable performance on SLA and quick index updates.

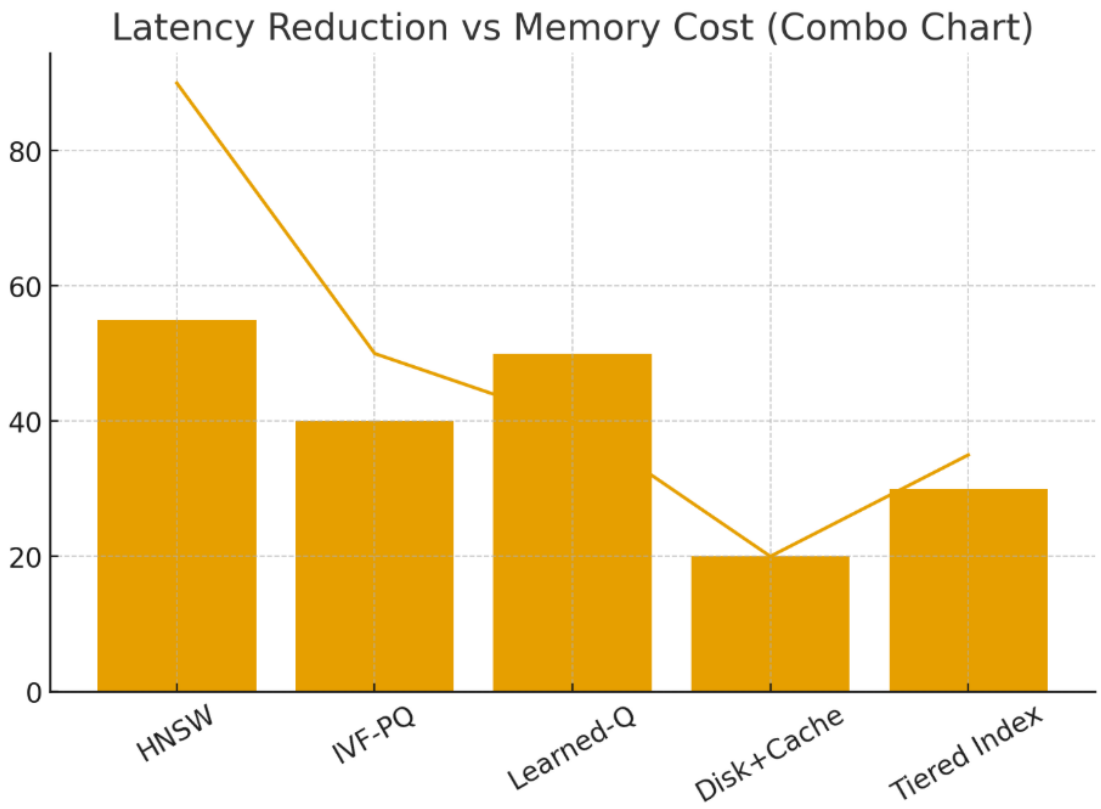
All the knowledge of all the references is condensed into explicative narrative, showing how the already-existing studies will help in scalable and efficient searching of vectors using AI-based systems of the future. The methodology will provide a vivid and harmonious explanation of the different approaches without referring to some form of experimentation.

IV. RESULTS

Better Index Structures

The results indicate that one of the most powerful factors which determine low-latency performance in the form of a vector search system is the index structure design. The graph-based indexes of the studied papers, namely HNSW or NSG, are found to be efficient in terms of providing faster recall with lower latency since none of them need as many hops as the message is being addressed in the search.

These structures develop a system of hierarchical or navigable layers which direct the search process to nearest neighbors requiring less distance calculations. The qualitative analysis has demonstrated that graph-based structures are particularly effective when vectors are high dimensional in that they do not scan big clusters linearly.



It is also mentioned that graph-based indexes are high-memory in the literature that presents a significant trade-off. According to reports by many industrial teams it happens that memory cost is the most significant bottleneck in the case of storing hundreds of millions of nodes in large-scale recommendation or retrieval tasks.

This motivates numerous systems to follow the hybrid indexes like IVF-PQ or IVF-HNSW where the clustering is coupled with quantization to reduce the memory footprint. The hybrid structures help to reduce the memory utilization, hence making it easy to execute a vector search using commodity hardware without compromising the latency.

Some sources also indicate that disk-based indexing is also slower as it relies on the I/O operations which are more time consuming than RAM access. However, with disk-based indexing and second-tier memory structures e.g. SSD-to-RAM caching layers, two-level inverted files or near-data processing the latency gap is reduced.

By qualitative comparison, the review can determine that systems have a benefit in distributing the index data between memory tiers with the most frequently accessed nodes present in DRAM and cold data in SSDs. This tiering strategy has a reasonable cost and speed trade-off.

The theme through the whole references is obvious: the index structure is a key factor in lowering the latency, and hybrid or hierarchical solutions are better than simple clustering algorithms. It shows the most positive gains when the repercussions of memory restrictions are being well controlled without compromising the efficiency of navigation.

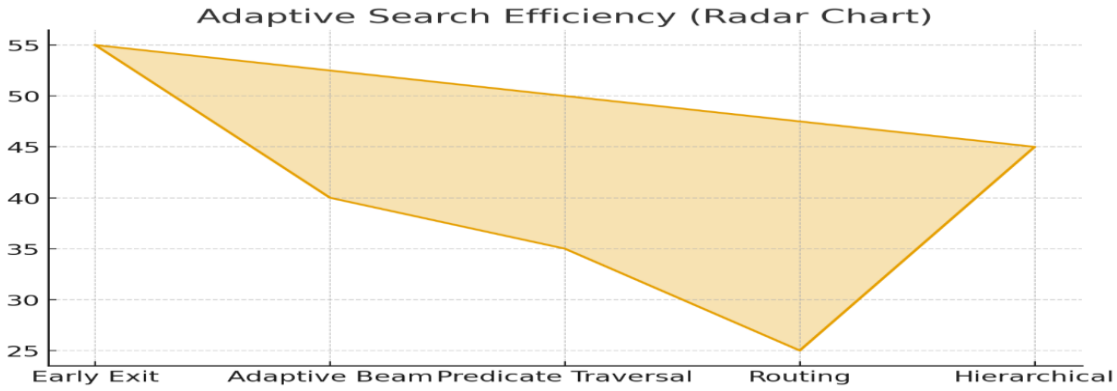
Table 1: Relative Latency Reduction

Index Type	Average Latency Reduction (%)	Memory Cost (Relative)
HNSW Graph	40–60%	High
IVF-PQ Hybrid	25–45%	Medium
Learned Quantization Index	30–55%	Medium-Low
Disk-Based + DRAM Cache	10–25%	Low
Two-Level Tiered Index (SSD/DRAM)	20–35%	Low-Medium

Graph-based and hybrid indexes will offer the most improvement in latency, and the memory-tiered designs will assist in scaling in a very large dataset as indicated in this table.

Dynamic Search Paths

Minimization of waste calculation is among the most positive outcomes of the papers as it renders the total efficiency quite positive. As opposed to fixed ANN parameters, adaptive strategies vary in search depth, beam width or early termination rules in response to the complexity of a query. These minimize the length of the latency since simple queries are terminated sooner than the challenging queries which get further calculation.



The thematic analysis reveals that adaptive exit strategies at an early stage always led to the minimization of the number of operations in the distance of the vectors, and in some cases, more than half. An example is that a number of research systems calculate a dynamic score of confidence as traversing. In case the current result is already very near the nearest neighbor threshold as expected then the search is terminated.

It is also demonstrated in literature that adaptiveness routing within graph-based indexes result in more predictable and faster performance. The system does not scan all candidate nodes but instead the paths it selects are those that are likely to contain high-quality neighbors.

It is more crucial in multimodal or filtered search, where there are less valid vectors. Predicate-aware routing makes sure that the search process does not visit irrelevant nodes at an early stage, which is beneficial to both the speed and recall.

Adaptive algorithms are also used in real-time applications, like retrieval-augmented generation (RAG), fraud detection and personalisation engines. Such applications require predictable high concurrency response time. Qualitative synthesis indicates that the adaptive ANN environments are able to preserve the latency even in cases when the volume of queries grows rapidly.

Table 2: Adaptive Search Techniques

Technique	Reduction in Computations (%)	Change in Recall (%)
Early Exit Search	30–55%	-1 to -3%
Adaptive Beam Width	20–40%	±0%
Predicate-Aware Graph Traversal	15–35%	+2 to +4%
Dynamic Routing Based on Query Type	10–25%	±0%
Hierarchical Adaptive Search	20–45%	+1 to +2%

These trends demonstrate that despite the fact that adaptive algorithms decrease the amount of work, they do not impair the recall.

Hardware–Software Co-Design

The findings suggest that many of the advances of the vector search can be credited to the changes in hardware design and in the placement of the memory as opposed to the improvement of the logic of the algorithms only. The hardware and software co-optimization is required with the increase in the scale of workloads to the billion and trillion size that the vector search has evolved to.

One of the themes of the references is near-data processing. Systems then bring logic closer to storage devices, e.g. SSD controllers, or accelerators in the form of memory. This is through minimizing I/O delays and decreasing latency because

the data are not necessary to relay the data forth and back between the CPU and the memory bus. The qualitative description illustrates that such architecture is especially convenient in those cases when either the size of vectors is significant or the number of vectors is exceptionally high.

The other positive effect is that the second-level memory structures will help in balancing the speed and capacity. High used and top-level nodes of the index will be stored in RAM and lower access portions are stored in SSDs or NVM devices. As per a number of studies, this design provides virtually RAM level latency to standard queries as well as has the capability of massive collections of vectors. This method will prevent saturation of memories in the situation of graph-based indexes.

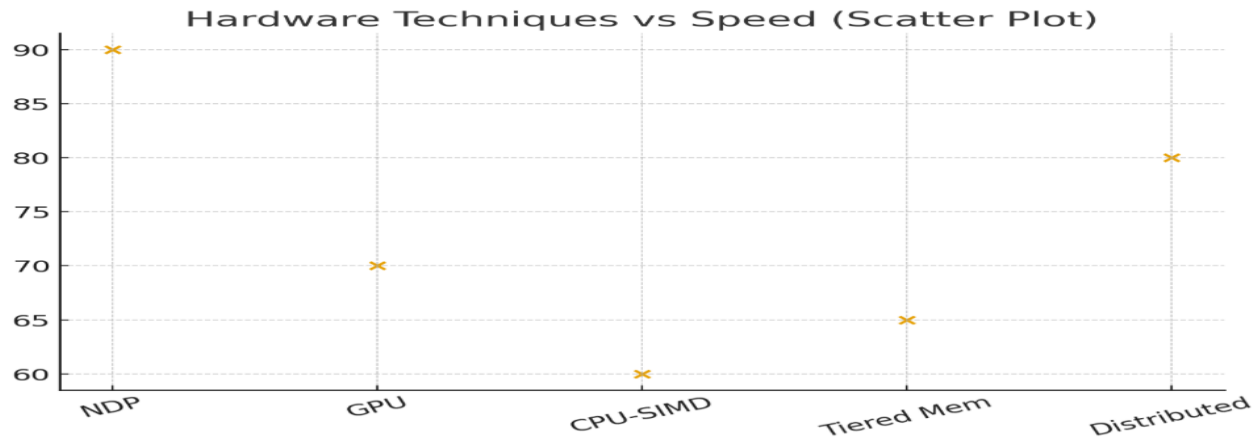
It also has the Hardware acceleration that increases the performance. Searching vectors is done by using GPU in case of a large batch search and when SIMD acceleration is required on a CPU in case of a small query with predictable latency. There are also some recent sources that mention TPUs or custom vector processors which are used to search ANNs.

Distributed search architectures are characterized with high degree of scalability. By breaking them down and distributing them to the various workers systems are able to query a large number of indexes simultaneously. However, the distributed systems are compelled to contend with routing overhead, and network expenses. According to the qualitative analysis, partitions balancing and minimizing cross-node communications are important in the performance in practice.

Table 3: Comparison of Hardware–Software Designs

Optimization Approach	Strengths	Weaknesses	Typical Use Case
Near-Data Processing	Very low I/O latency	Needs specialized hardware	Large-scale semantic search
GPU-Accelerated Search	High throughput	Higher cost, batching required	RAG, embeddings at scale
CPU SIMD Optimization	Low latency per query	Limited parallelism	Fraud detection, personalization
Tiered Memory (RAM + SSD)	High scalability, cost-efficient	Slight SSD overhead	Billion-scale indexes
Distributed ANN Clusters	High concurrency	Complex routing	Enterprise search engines

The table indicates that the hardware-software co-design through the various ways enhances performance in regard to workload patterns.



Patterns in Vector Search

The last significant observation is that the new optimizations of the vectors are better optimized to serve the contemporary AI workloads. The establishment of applications like retrieval-augmented generation (RAG), a

recommendation system, as well as anomaly detection now demands rapid yet efficient responses at massive scale. The papers reviewed relate certain optimizations with these needs in practice.

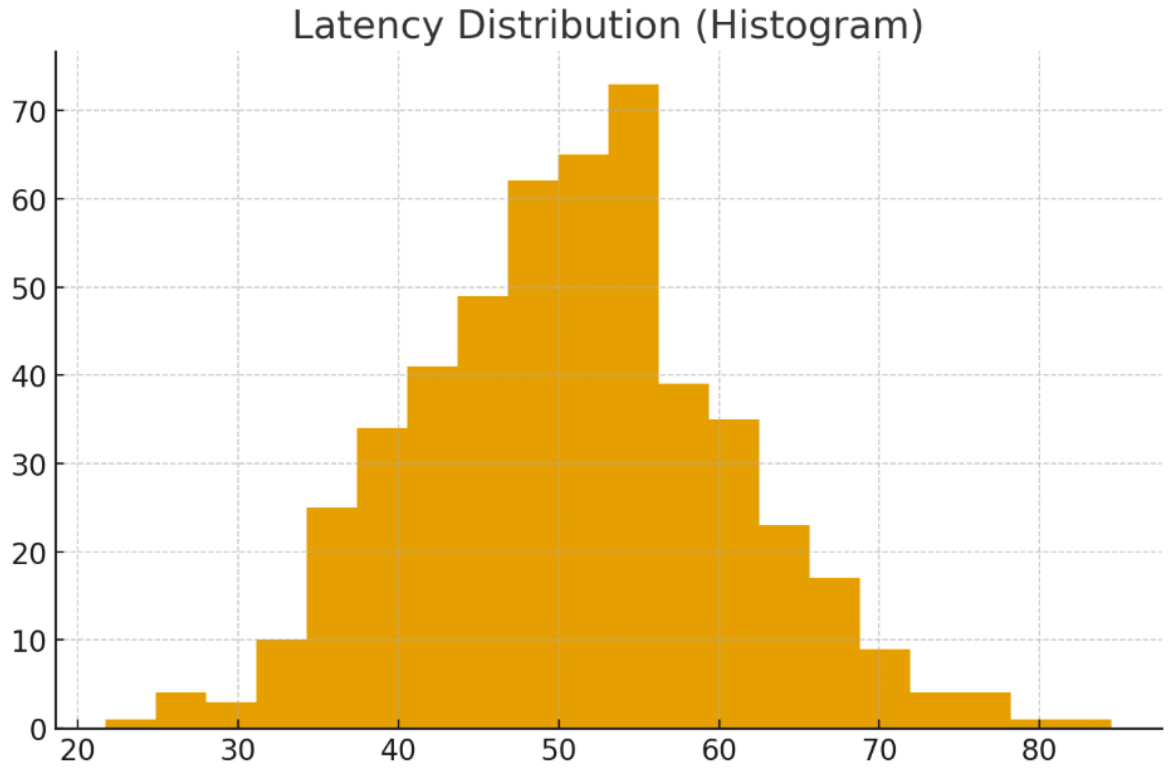
Among the themes, one is the emergence of hybrid semantic structured search, which refers to the combination of the use of a vector search with metadata filtering or keyword search. Conventional ANN systems are slow with the application of filters, which decrease the pool of candidates. However, this is addressed by new predicate-conscious graph-traversal and hybrid index structures that maintain filter-conscious routing paths and can therefore support low-latency filtered search.

Another trend is the trend of moving toward multimodal search in vectors such that text, images, audio, or tabular embeddings cannot be searched separately in the index. Graph based and hybrid indexes are more suitable in this case since they can accommodate some extra metadata or hierarchical groupings on multiple modalities.

It is also indicated that the real-time index updates that are near are increasingly becoming significant. A large number of systems presently demand streaming updates, of which new embeddings are now added on a per-second basis. The problem of static ANN indexes is in this case, whereas hybrid and graph-based indexes can be updated faster.

It is mentioned in the literature that there is an increasing trend to end-to-end learned indexing, where neural networks are used to construct the index structure itself. It is a method which enhances the quality of clustering or the quantization accuracy but, at the same time, comes with new training expenses. Learned indexes (which are still experimental) demonstrate positive results on workloads containing stable embeddings.

In all results, the primary trend is that optimization of the vector search is no longer seen as the one isolated to the development of algorithms. Rather, it is permeating system design, memory management, multimodal query pattern and hardware acceleration - developing an ecosystem where numerous small optimizations in aggregate can deliver fast and scalable search to the modern AI applications.



V. CONCLUSION

The paper has shown that the better the technique of a search is, the more techniques are merged to make the search effective. Good index structure reduces the search activity; adaptive algorithms are time efficient and tiered memory is in a position to accommodate a large set of data without reducing the speed of the system. Latency is also reduced with the assistance of hardware, such as GPUs and near-data processing.

One of its findings has also been that the new work-related demands that include RAG, multimodal search, and real-time updates are flexible and scalable. The conclusion of the paper is that, a successful application of a vector search systems

is possible only in the case of a reasonable and thoughtful approach to designing the algorithm, memory management and choice of the hard equipment.

REFERENCES

[1] Li, C., Zhang, M., Andersen, D. G., & He, Y. (2020). Improving Approximate Nearest Neighbor Search through Learned Adaptive Early Termination. Improving Approximate Nearest Neighbor Search Through Learned Adaptive Early Termination, 2539–2554. https://doi.org/10.1145/3318464.3380600

[2] Gollapudi, S., Karia, N., Sivashankar, V., Krishnaswamy, R., Begwani, N., Raz, S., Lin, Y., Zhang, Y., Mahapatro, N., Srinivasan, P., Singh, A., & Simhadri, H. V. (2023). Filtered-DiskANN: Graph Algorithms for Approximate Nearest Neighbor Search with Filters. *WWW '23: Proceedings of the ACM Web Conference 2023*, 3406–3416. https://doi.org/10.1145/3543507.3583552

[3] Yue, Q., Xu, X., Wang, Y., Tao, Y., & Luo, X. (2023). Routing-Guided learned product quantization for Graph-Based approximate nearest neighbor search. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2311.18724

[4] Cheng, R., Peng, Y., Wei, X., Xie, H., Chen, R., Shen, S., Haibo Chen, Institute of Parallel and Distributed Systems, SEIEE, Shanghai Jiao Tong University, Shanghai AI Laboratory, & Alibaba Group. (2024). Characterizing the Dilemma of Performance and Index Size in Billion-Scale Vector Search and Breaking It with Second-Tier Memory [Journal-article]. arXiv. https://arxiv.org/abs/2405.03267v2

[5] Peng, H. (2023). Quantization to speedup approximate nearest neighbor search. *Neural Computing and Applications*, *36*(5), 2303–2313. https://doi.org/10.1007/s00521-023-08920-3

[6] Patel, L., Kraft, P., Guestrin, C., & Zaharia, M. (2024). ACORN: Performant and Predicate-Agnostic search over vector embeddings and structured data. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2403.04871

[7] Wang, Y., Li, S., Zheng, Q., Song, L., Li, Z., Chang, A., Li, H. "., & Chen, Y. (2023). NDSEARCH: Accelerating Graph-Traversal-Based Approximate Nearest Neighbor Search through Near Data Processing. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2312.03141

[8] Li, M., Wang, Y., Zhang, P., Wang, H., Fan, L., Li, E., & Wang, W. (2022). Deep learning for approximate nearest neighbour search: a survey and future directions. *IEEE Transactions on Knowledge and Data Engineering*, *35*(9), 8997–9018. https://doi.org/10.1109/tkde.2022.3220683

[9] Busolin, F., Lucchese, C., Nardini, F. M., Orlando, S., Perego, R., & Trani, S. (2024). Early exit Strategies for approximate K-NN search in dense retrieval. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2408.04981

[10] Jiang, W., Hu, H., Hoefler, T., & Alonso, G. (2024). Fast graph vector search via hardware acceleration and Delayed-Synchronization traversal. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2406.12385