# Towards Optimizing Search and Indexing in Vector Databases

Jayjeet Chakraborty (jayjeetc@ucsc.edu) , Heiner Litz (hlitz@ucsc.edu)

Center for Research in Systems and Storage

UNIVERSITY OF CALIFORNIA SANTA CRUZ

## Motivation

Vector searches sit on the critical path while querying RAG applications, and hence getting fast responses need vector searches to be highly performant. Also, in hyper-scale setups, vector datasets contain billions of points and require modern hardware for indexing and search operations.

Therefore, having an in-depth understanding of the performance characteristics of vector indexing and search algorithms is crucial.

## Vector Search and Indexing

Since, kNN algorithms scale linearly and result in unrealistic search times, ANN algorithms were developed which trade some accuracy for search speed. At the heart of ANN algorithms, are index structures that help narrow down the search space to enable faster searches. Some popular vector indexing algorithms consist of:

- Flat Index
- Inverted File Index (IVF)
- Hierarchical Navigable Small Worlds (HNSW)
- Locality-Sensitive Hashing (LSH)
- Vamana / DiskANN

## Vector Databases / Libraries

Milvus

qdrant

LanceDB

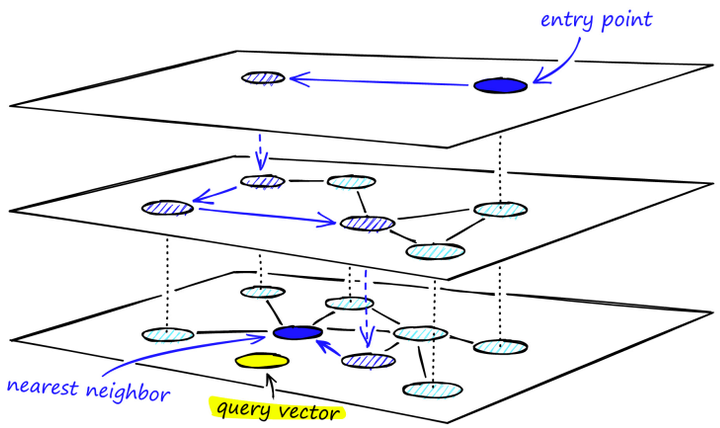
Weaviate

Chroma

nmslib / hnswlib

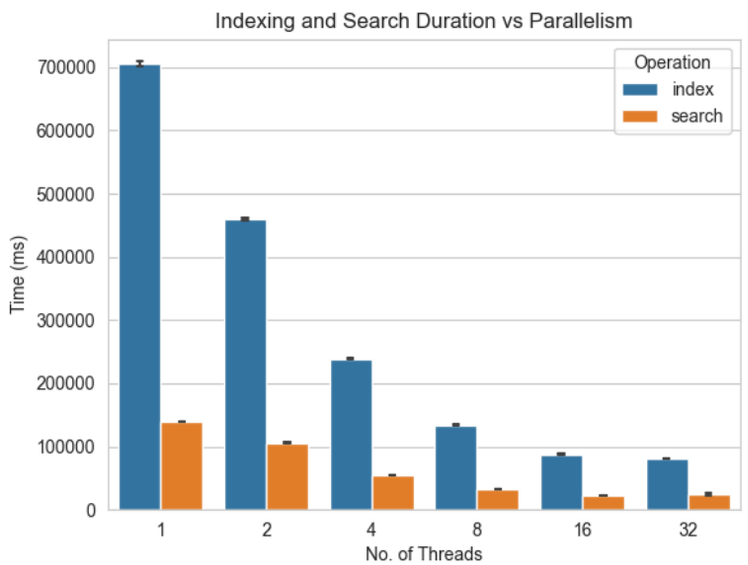
facebookresearch / faiss

## Hierachical Navigable Small World Graphs




We start from layer 0, pick a entry node, compare its neighbors with the query vector, and move to the neighbor closest to the query vector. Once we find a local minima, we move to that exact node in the next layer and start the search again. The local minima that we find in the last layer is the closest one to our query vector.

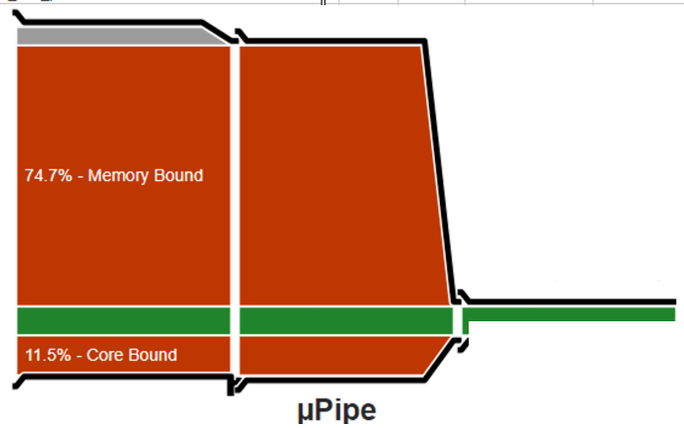
## Effect of Parallelism in hnswlib




## Profiling Vector Searches in hnswlib

| Source Function / Function / Call Stack | CPU Time ▼ |
|---|---|
| _mm_sub_ps | 91.072s |
| hnswlib::L2SqrSIMD16ExtSSE | 29.236s |
| _mm_loadu_ps | 17.870s |
| hnswlib::HierarchicalNSW<float>::searchBaseLayerST<(bool)1, (bool)0> | 14.147s |
| _mm_add_ps | 7.778s |
| _mm_mul_ps | 2.438s |
| _mm_prefetch | 0.824s |
| hnswlib::HierarchicalNSW<float>::searchKnn | 0.670s |

| Source Function / Function / Call Stack | | Back-End | | | |
|---|---|---|---|---|---|
| | | L2 Bound | L3 Bound | DRAM Bound | |
| | | | | Memory Bandwidth | Memory Latency ▼ |
| _mm_loadu_ps | % | 25.2% | 0.9% | 59.6% | 32.4% |
| page_fault | % | 31.5% | 0.0% | 0.0% | 31.8% |
| _mm_add_ps | % | 28.1% | 2.6% | 61.1% | 25.5% |
| _mm_prefetch | % | 0.2% | 21.3% | 39.7% | 25.3% |
| _mm_sub_ps | % | 72.7% | 5.8% | 77.2% | 17.0% |
| hnswlib::L2SqrSIMD16ExtSSE | | 75.2% | 3.7% | 79.0% | 15.0% |
| call_function_interrupt | % | 7.3% | 0.0% | 24.0% | 14.5% |
| hnswlib::HierarchicalNSW<float>::searchBaseLayer | % | 0.4% | 29.8% | 85.4% | 14.4% |
| _mm_mul_ps | % | 52.1% | 1.3% | 61.4% | 11.8% |




## Conclusion

Vector search operations can't be batched and can only be parallelized

In-depth understanding of the performance characteristics of vector searches is crucial to find opportunities for acceleration using modern hardware

Vector searches are mostly memory bandwidth bound and most of the query duration is dominated by distance calculations.