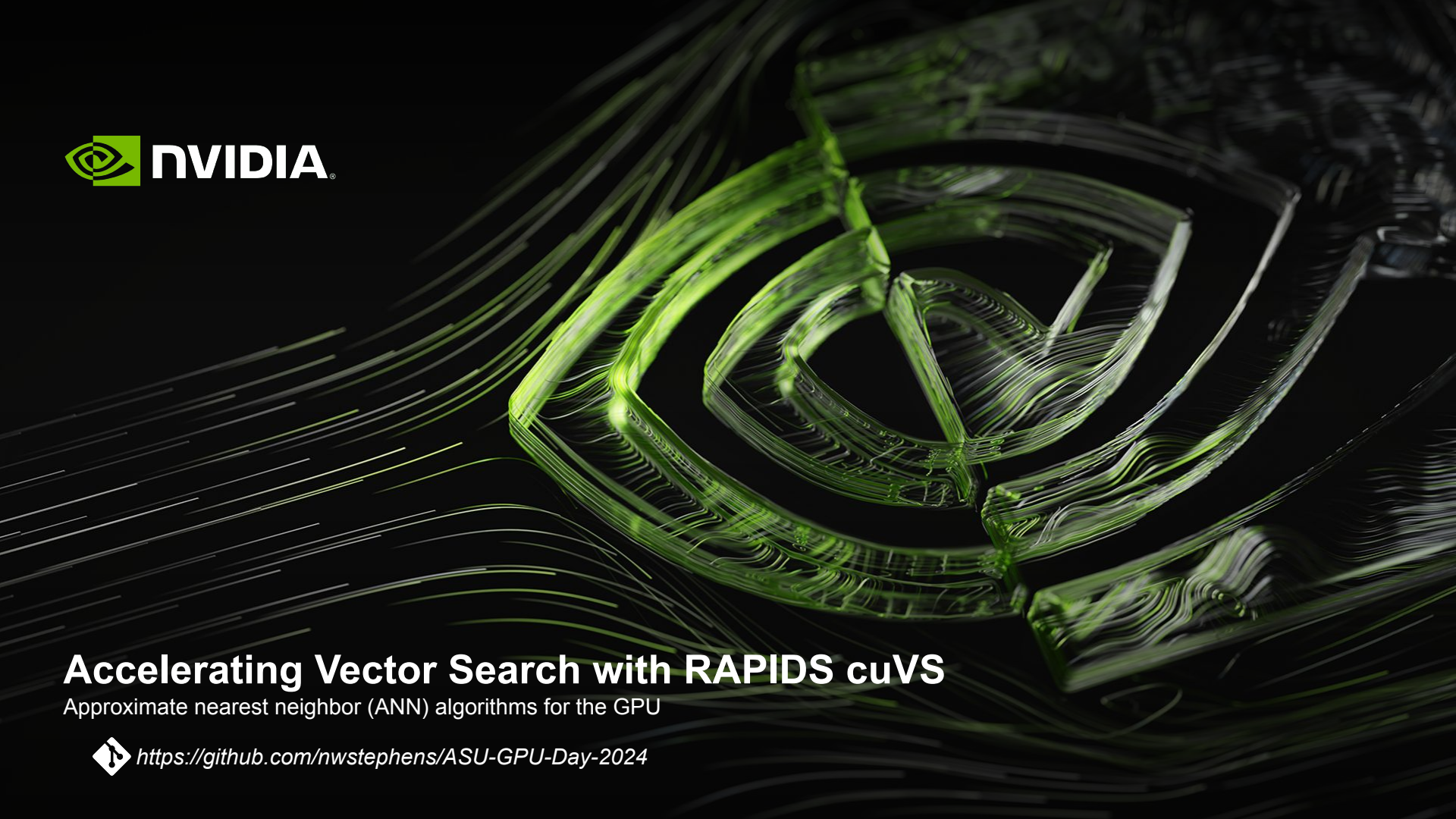


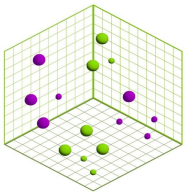
Accelerating Vector Search with RAPIDS cuVS

Approximate nearest neighbor (ANN) algorithms for the GPU

https://github.com/nwstephens/ASU-GPU-Day-2024

Contents

Vector Search



Overview

Embeddings

Vector Search

Applications

Vector Search Algorithms

IVF-Flat

IVF-PQ

CAGRA

RAPIDS

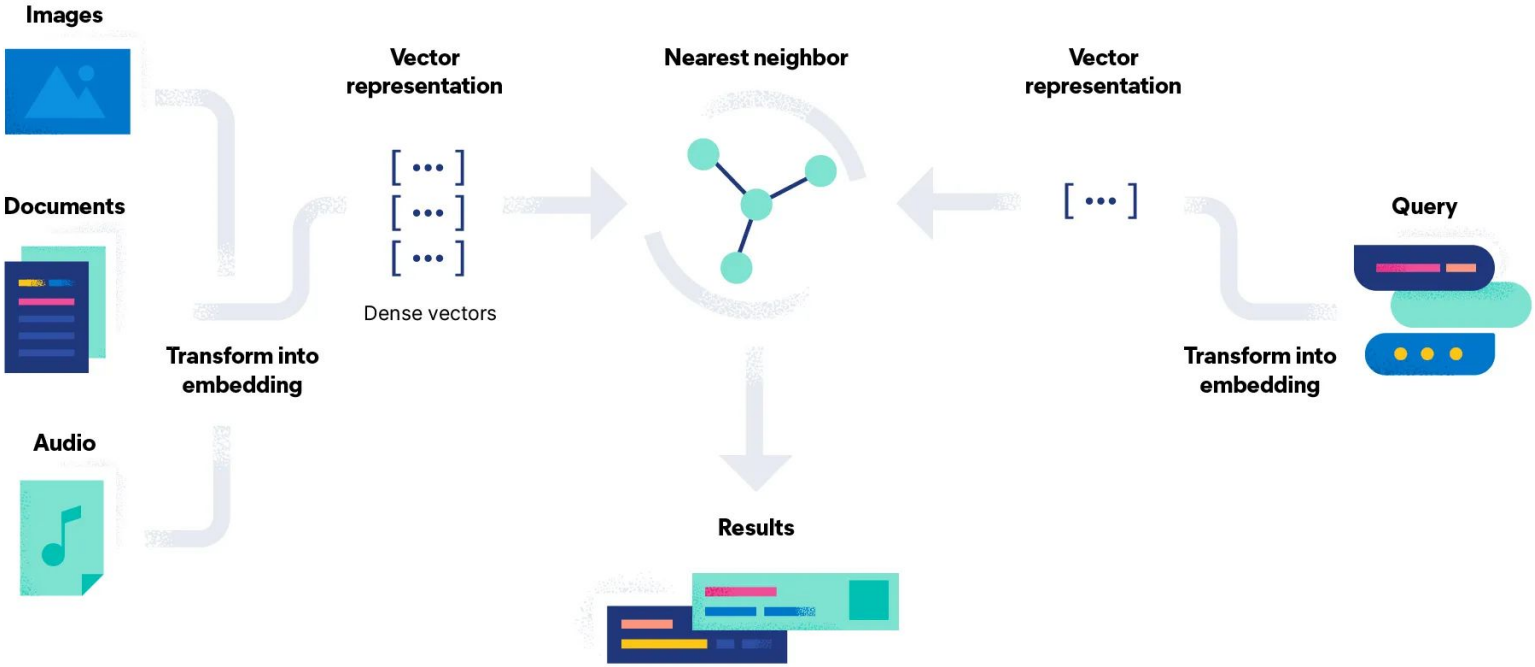
Notebooks

jupyter

Vector Search Overview

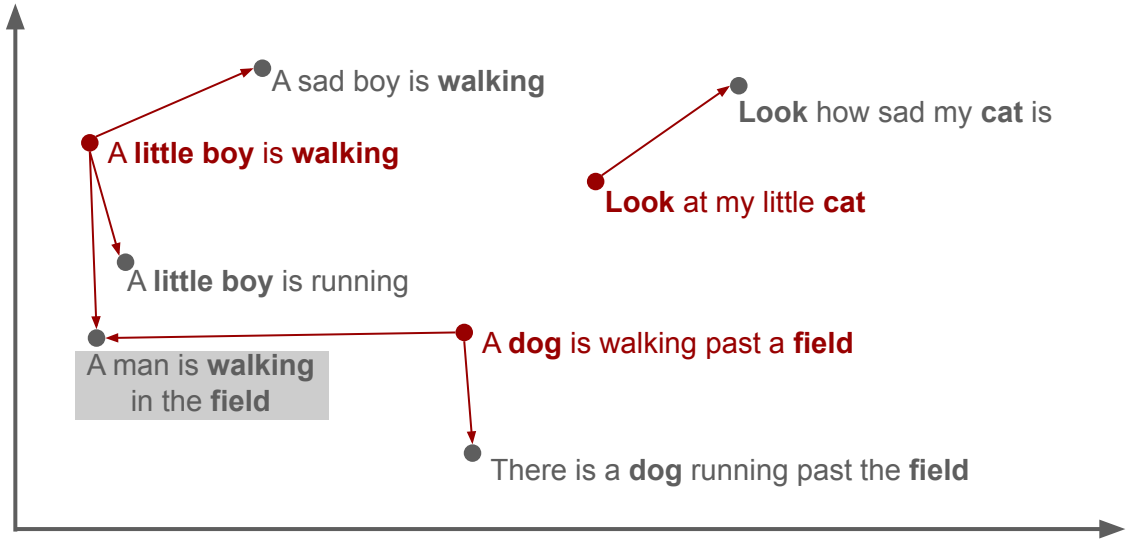
Vector Search Overview

Nearest Neighbor Analysis



Vector Search Overview

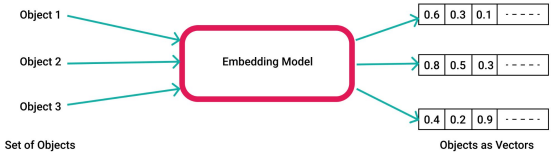
Example of Text Vectors

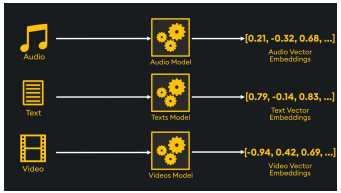


Vector Search Overview

What is an embedding?

Embedding: Numerical representation of typically non-numerical data objects





Audio

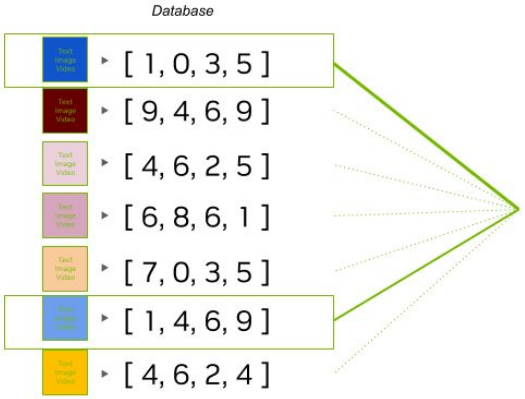
- Spectrogram-based Representations
- MFCCs (Mel Frequency Cepstral Coefficients)
- Convolutional Recurrent Neural Networks (CRNNs)

Text

- TF-IDF (Term Frequency — Inverse Document Frequency)
- Word2Vec
- BERT (Bidirectional Encoder Representations from Transformers)

Image

- Convolutional Neural Networks
- Transfer Learning with Pre-trained CNNs like ResNet and VGG
- Autoencoders

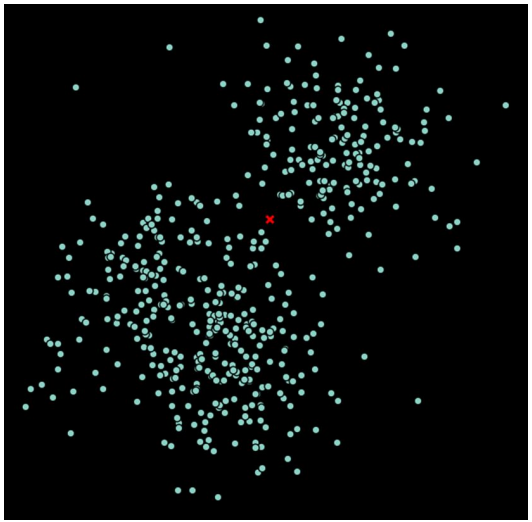


"Find the two most similar images, documents, or videos"

Vector Search Overview

Nearest Neighbor Search

Question: How to find the closest k points to the red marker?



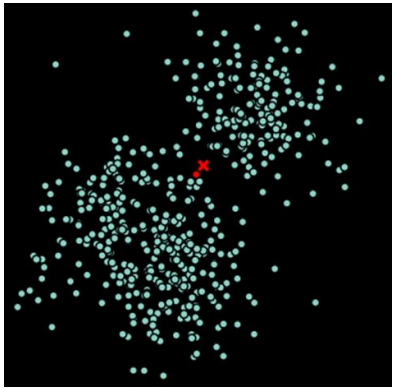
Vector Search Overview

Nearest Neighbor Search

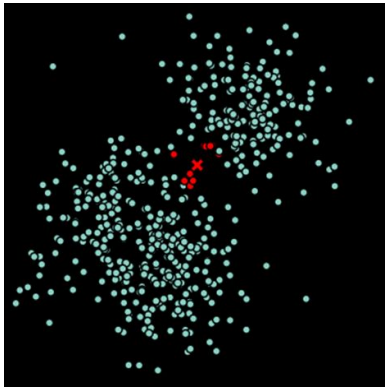
Question: How to find the closest k points to the red marker?

Answer: Measure the distance to all points and choose the shortest!

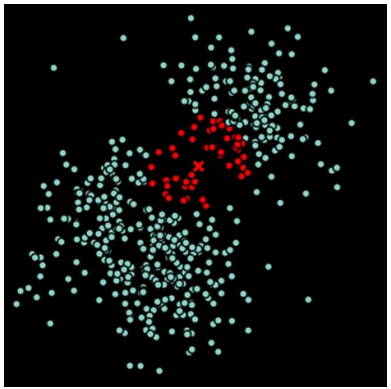
K=1



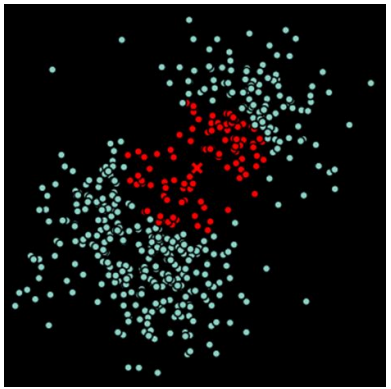
K=10



K=50

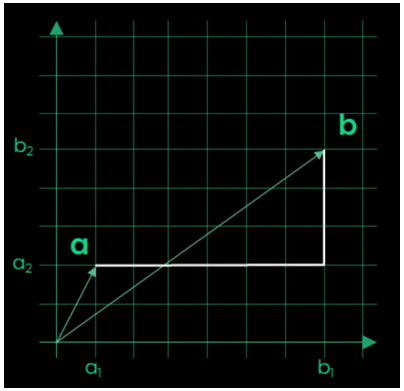


K=100

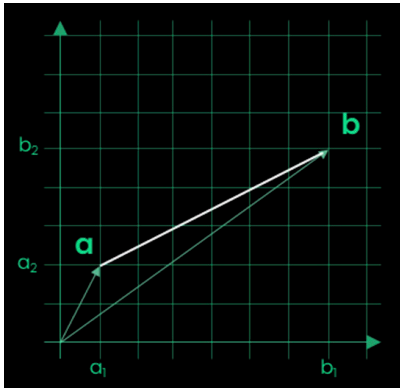


Vector Search Overview

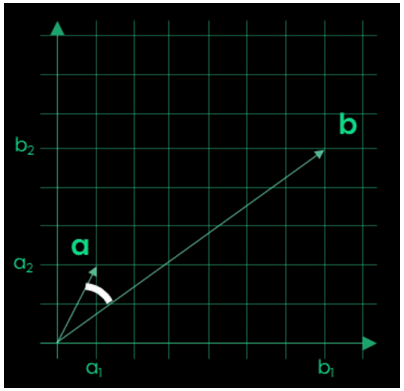
Example Distance Metrics



L1
(Taxi Distance)



L2 or Euclidean
Distance



Cosine
Similarity

Vector Search Overview

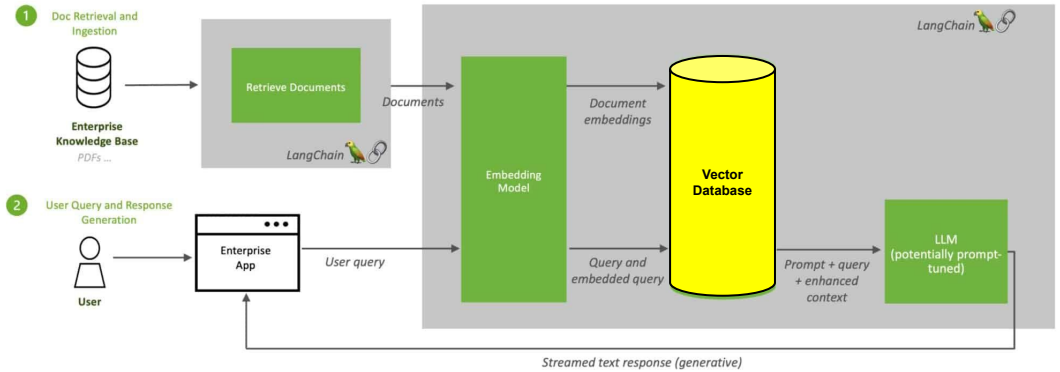
Applications

- **Recommender systems**. Provide personalized suggestions according to what a user has shown interest in or interacted with.
- **Finance.** Fraud detection models vectorize user transactions, making it possible to determine whether those transactions are similar to typical fraudulent activities.
- **Cybersecurity**. Uses embeddings to model and search behaviors of bad actors and anomalous activities.
- **Genomics.** Finds similar genes and cell structures in genomics analysis, such as single-cell RNA analysis.
- **Chemistry.** Models molecular descriptors or fingerprints of chemical structures to compare them or find similar structures in a database.

Vector Search Overview

Retrieval Augmented Generation (RAG)

RAG Workflow



Vector Search Technologies

Milvus

redis kinetica

Weaviate DATASTAX

APACHE LUCENE

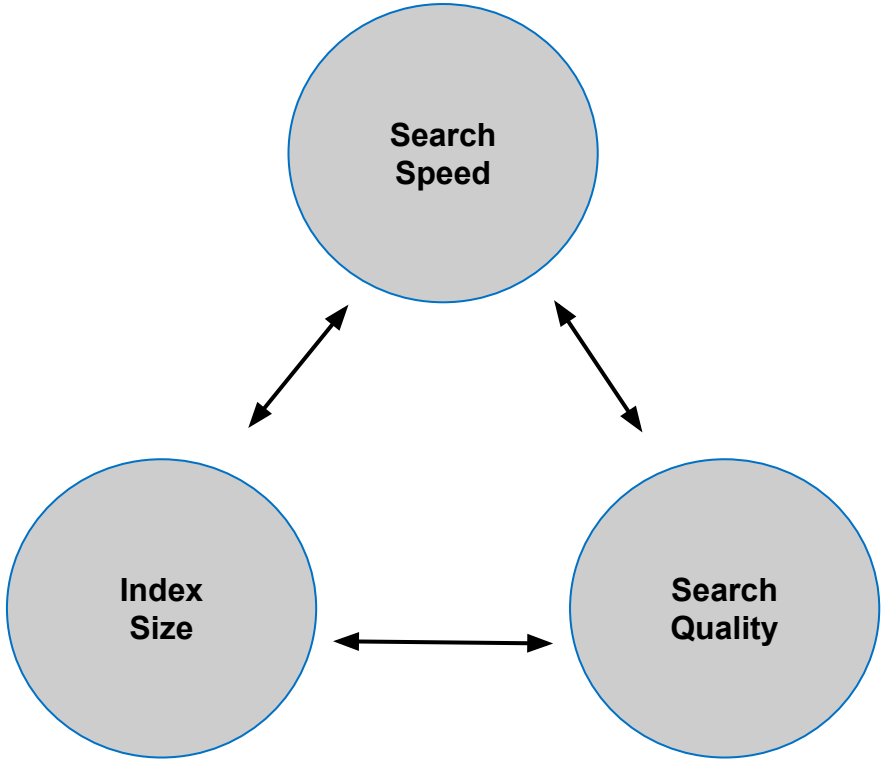
OpenSearch Solr

Vector Search Algorithms

Vector Search Challenges

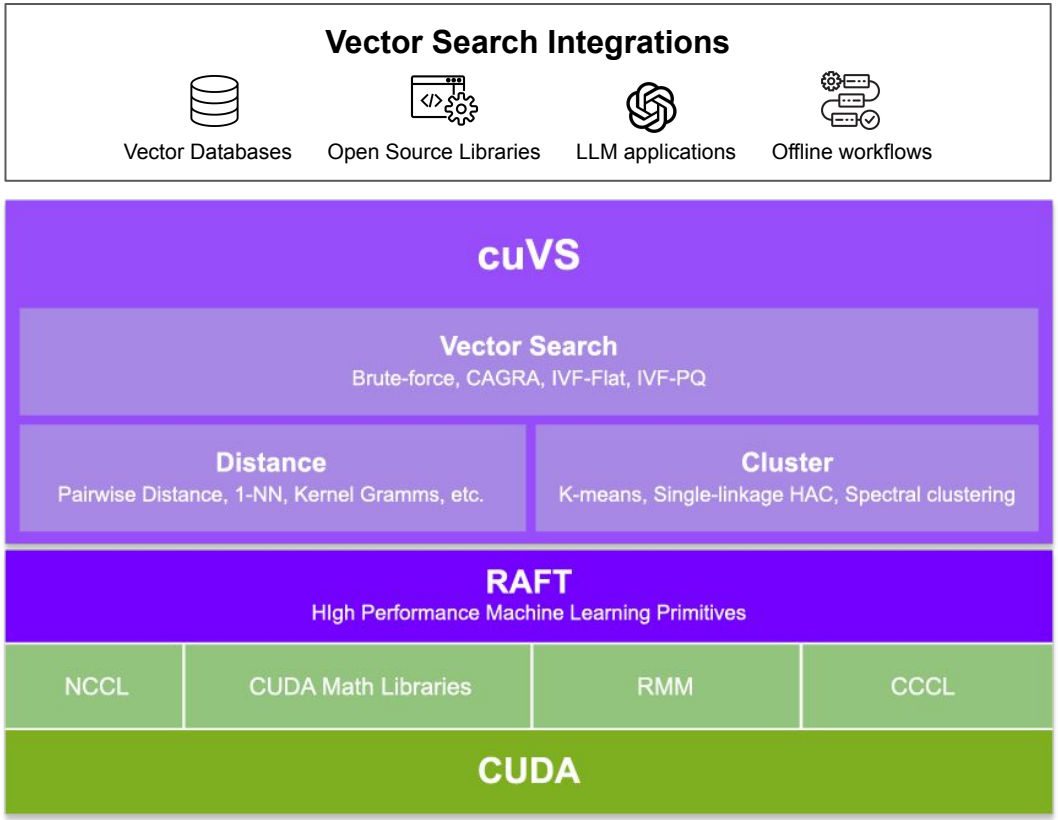
Competing priorities cause computational challenges and tradeoffs

- Higher throughput requires larger batch sizes but latency requirements may require small batch sizes.
- Indexing is slow and choice of index affects search quality, size, and speed.
- Higher quality search requires larger storage and is more thorough, reducing speed.
- Higher dimensionality embeddings can improve quality at the expense of storage and speed.
- Larger datasets increase computational needs, reducing speeds and increasing storage size.



RAPIDS cuVS Overview

GPU-Accelerated Vector Search



Approximate Nearest Neighbors

Building an Index

Brute force is slooooowwww for large data

- **Solution**: Build an index!
- **Examples**: Inverted file index (IVF) or graph based (HNSW)
- Gives **approximate** results (less than 100% recall)

Building an index

- Used for **efficiently querying** several vectors at a time.
- To handle **larger datasets** efficiently
- **Speed up** the search by approximating the closest vectors
- To ensure both fast searches and low memory usage, you must index vectors in an **efficient** way
- This can sometimes benefit from **compression**

Great for GPUs

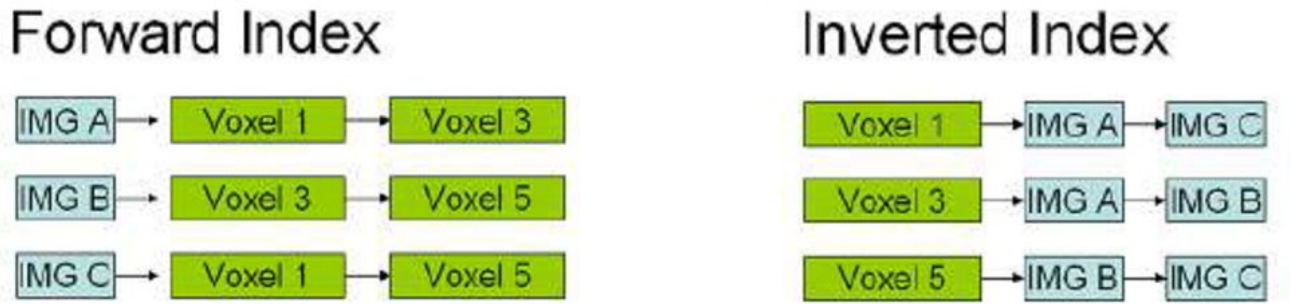
- Inserting and deleting vectors, can cause problems when indexes take hours or even **days to build on the CPU**.
- **These indexes can often be built much faster on the GPU.**

IVF-Flat

Inverted File Index

IVF-Flat

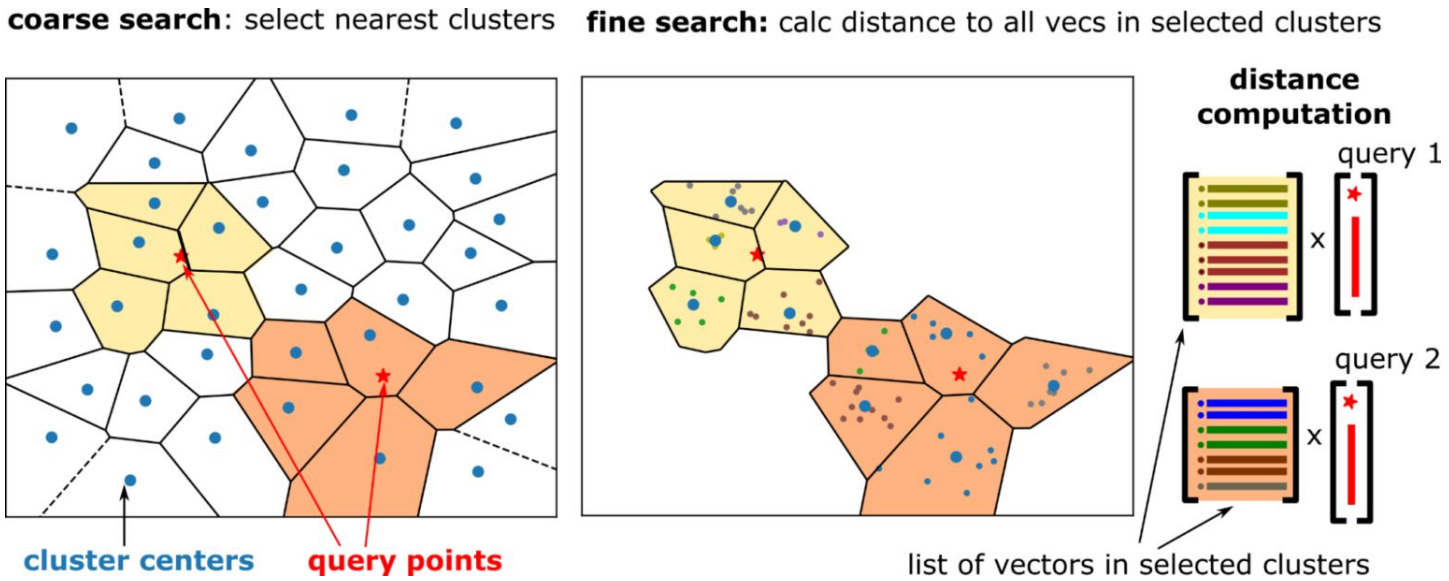
- Flat == unmodified vectors (i.e. no compression)
- Simple knobs to reduce the overall search space
- Trade-off accuracy for speed.



IVF-Flat

Coarse Search and Fine Search

The top-k neighbors from each probed cluster are selected, which results in **n_probes * k_neighbor** candidates for each query. This is reduced to the k-nearest neighbors.



IVF-Flat

Tuning parameters for index building

Tuning n_lists parameter

- **n_lists** parameter determines the number of clusters to partition the training dataset.
- **n_probes** determines the number of closest clusters to search through to compute the nearest neighbors for a set of query points.
  n_lists == sqrt(n_samples) is a good starting point (where n_samples is the number of vectors in the dataset).
- n_lists == n_probes is like an exact (brute force) search

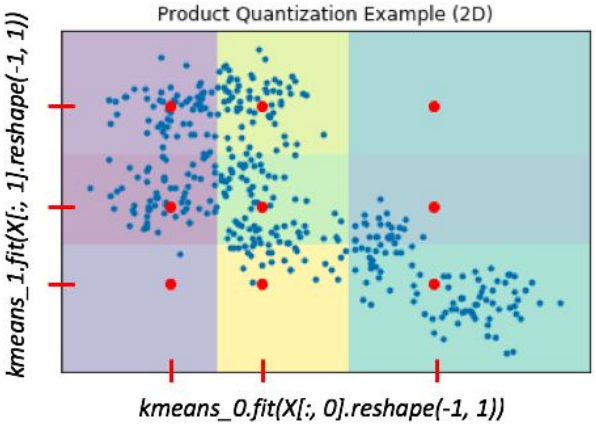
Optimizations

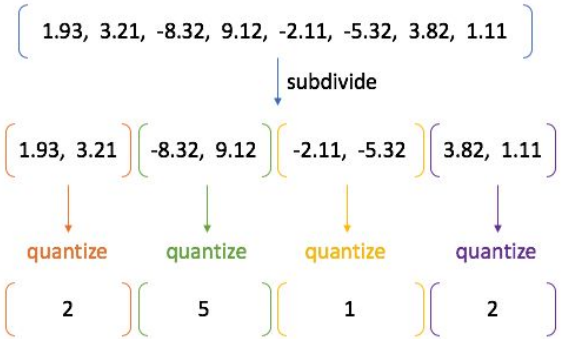
- **High throughput**. cuVS uses Tensor Cores to accelerate the k-means clustering during index building.
- **Improved algorithm**. cuVS uses a balanced hierarchical k-means clustering, which clusters the dataset efficiently even as the number of vectors reaches hundreds of millions.
- **Optimized top-k candidates**. cuVS has highly optimized methods to select the top-k candidates.

IVF-PQ

Product Quantization

- Lower PQ bits (4-8) provide ***better compression*** and more ***efficient use of shared memory***
- Configurable lookup table and distance precision provide ***faster computation*** and ***efficient use of shared memory***
- Support for ***reduced precision*** (uint8 and int8)
- Supports custom predicate ***pre-filter***
- ***pq_dim*** sets the target dimensionality of the compressed vector.
- ***pq_bits*** sets the number of bits for each vector element after compression.

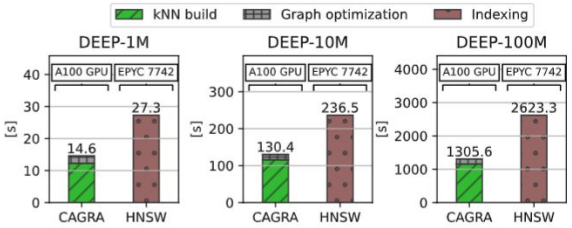


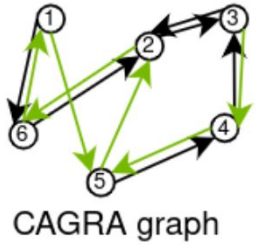


CAGRA

GPU-Accelerated State-of-the-Art Graph-Based ANN

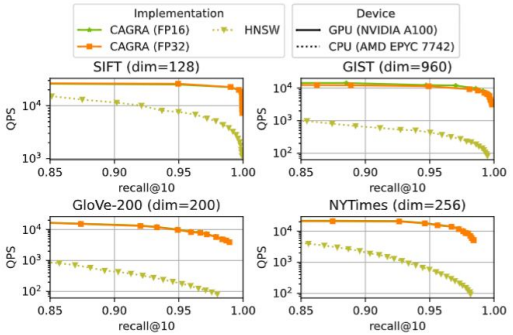
- ***Individual queries*** parallelized during search
- Setting records for both ***single query*** and ***large batch*** performance
- ***Higher throughput*** than existing GPU Graph ANNs and ***lower latency*** than SOTA CPU Graph ANNs



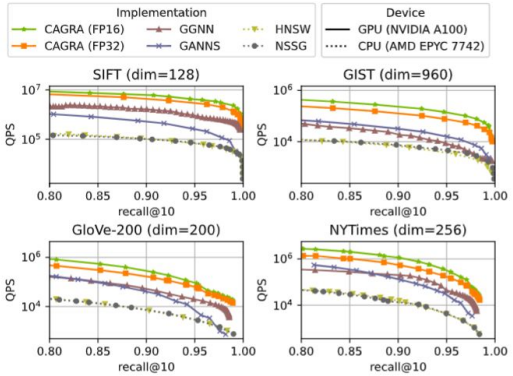


CAGRA graph

Single query at a time



Batches of 10k queries

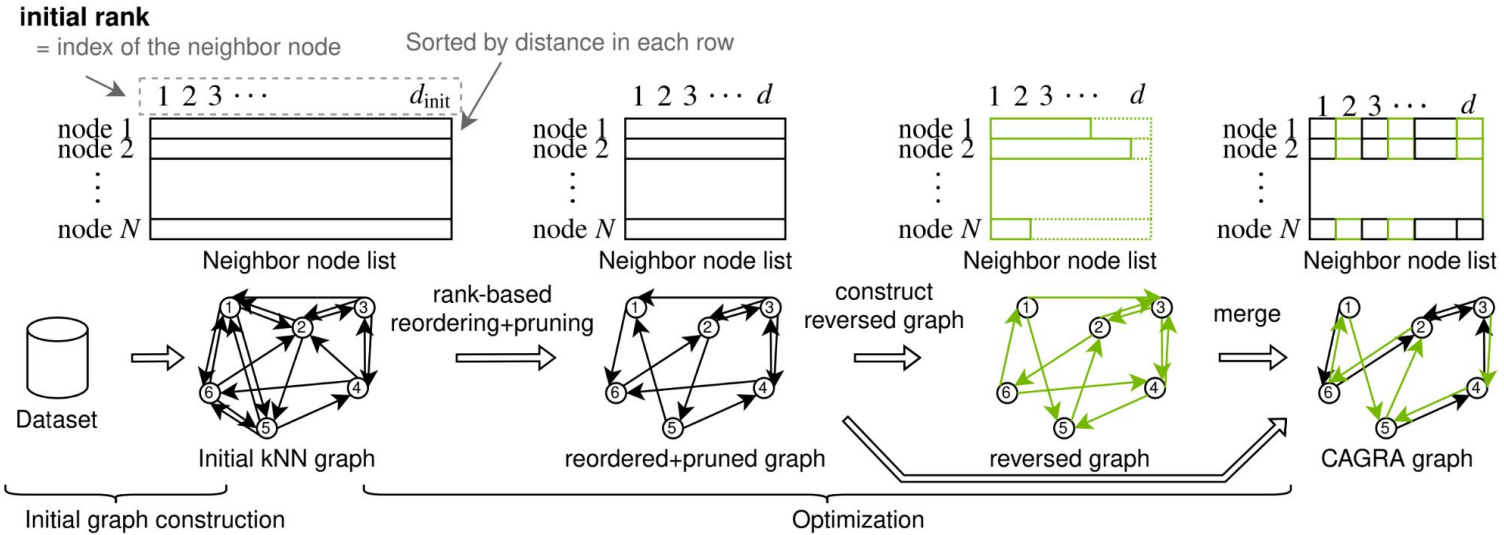


"CAGRA: Highly Parallel Graph Construction and Approximate Nearest Neighbor Search on the GPU", Ootomo et al., 2023

GPU Only

CAGRA

GPU-Accelerated State-of-the-Art Graph-Based ANN



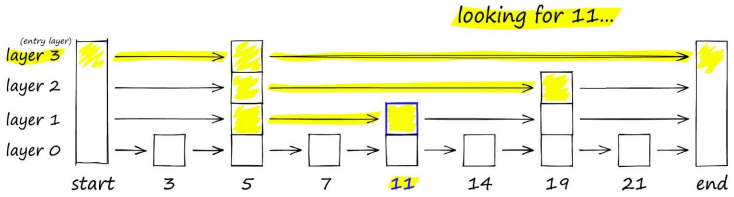
"CAGRA: Highly Parallel Graph Construction and Approximate Nearest Neighbor Search on the GPU", Ootomo et al., 2023

CPU Only

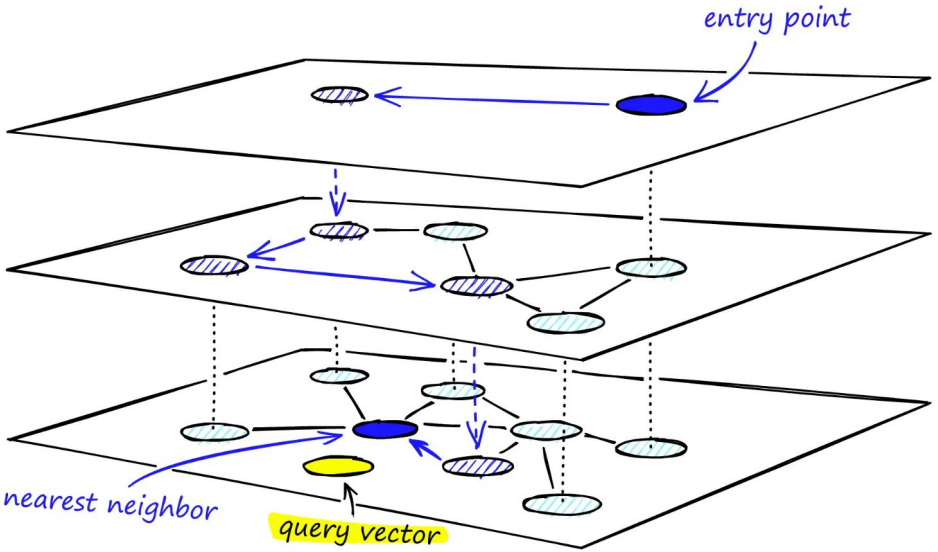
HNSW Overview

Hierarchical Navigable Small World (HNSW) algorithm

Skip List

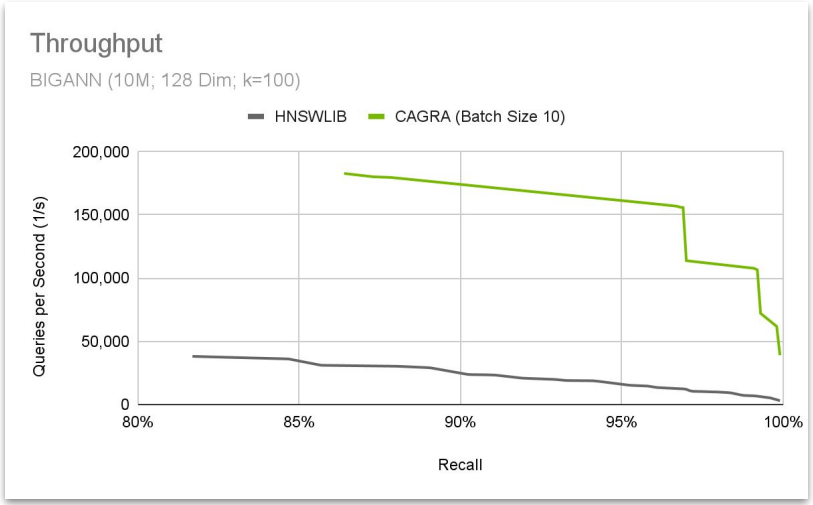


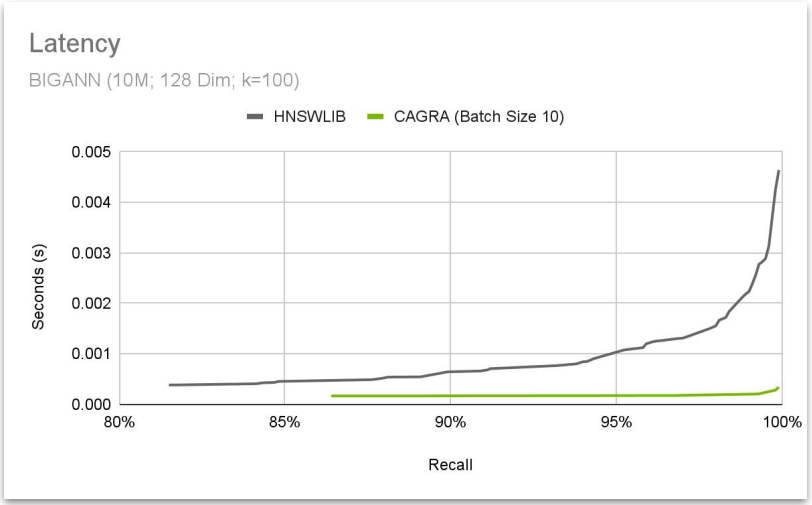
HNSW Graph



CAGRA (GPU) vs. HNSW (CPU)

Small batches (bs=10): Low latencies drive higher throughput



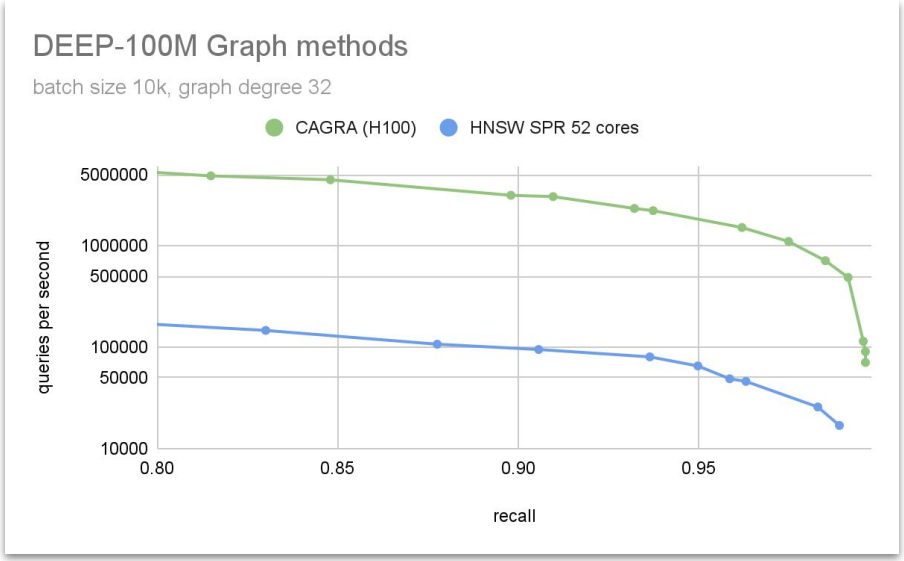


Recall	CAGRA	HNSW	Speedup
90%	174,153	24,816	**7.0X**
95%	161,356	16,008	**10.1X**
99%	108,029	6,940	**15.6X**

Recall	CAGRA	HNSW	Speedup
90%	0.000171	0.000647	**3.8X**
95%	0.000176	0.001035	**5.9X**
99%	0.000204	0.002236	**11.0X**

CAGRA (GPU) vs. HNSW (CPU)

Large batches (bs=10k): Increased parallelism drives larger speedups



Recall	CAGRA	HNSW	Speedup
90%	3,133,965	97,451	**32X**
95%	1,861,029	64,988	**29X**
99%	574,209	16,939	**34X**

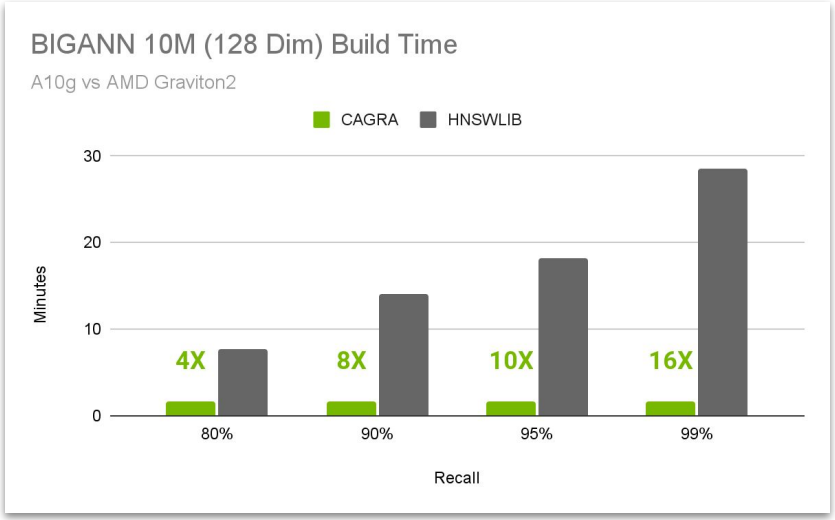
CAGRA Index Build Times

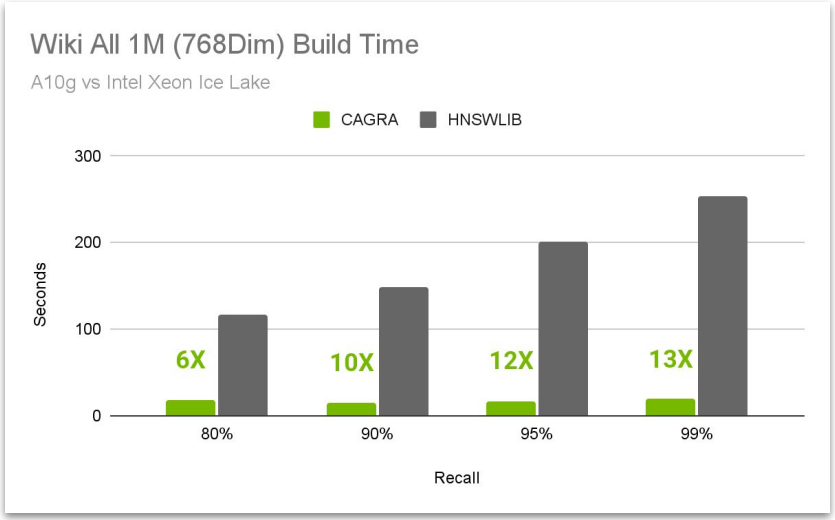
Compared to HNSW

GPUs are especially good at offline processes where large batch sizes can be used

Index build time increases as:

- Recall increases
- Dimensions increases
- Embeddings increase



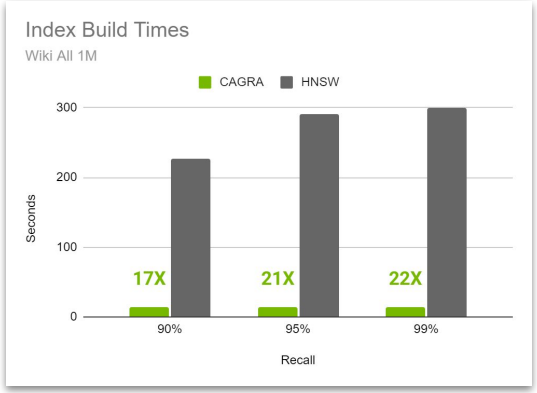


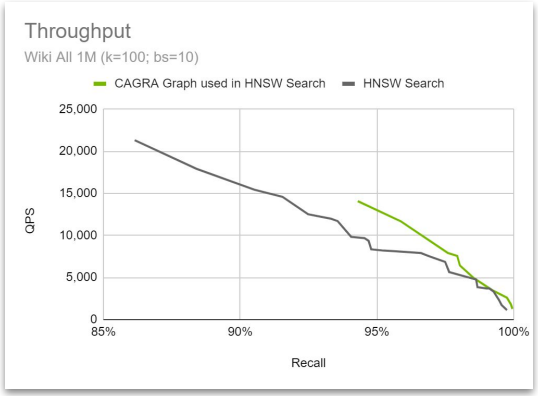
CAGRA+HNSW

Building index on GPU and searching on CPU

Interoperability between CPU and GPU

- Building HNSW indexes is slow – can take hours or days
- CAGRA indexes can be built 20x faster than HNSW
- HNSW can search a graph built with CAGRA
- CAGRA graph can even outperform HNSW on CPU search at larger dimensions

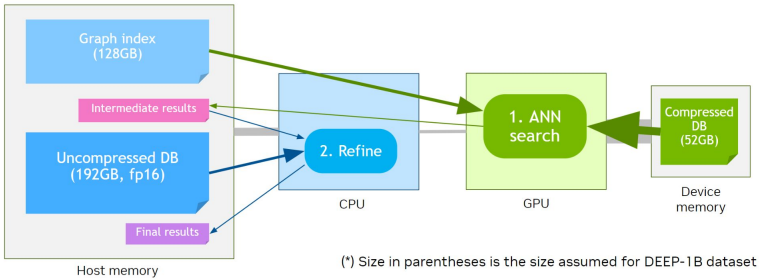




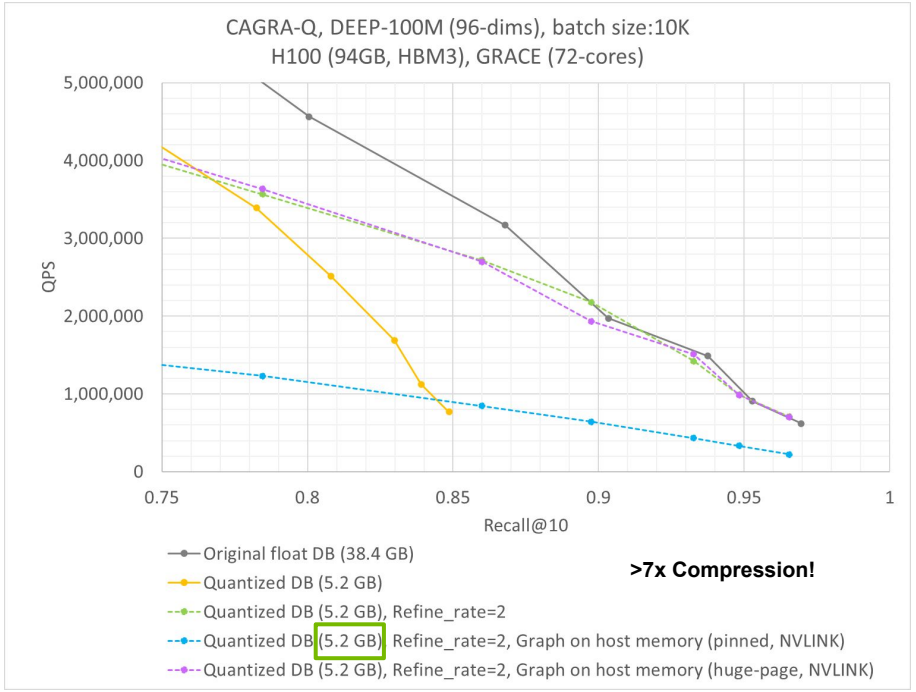
CAGRA-Q

CAGRA + Quantization for improved scale

- CAGRA requires ***original training vectors*** to compute distances
- Can keep original dataset in ***host memory*** (this can be slow)
- CAGRA-Q ***compresses original dataset*** so it can be stored on device for faster search
- Original dataset kept in host memory and used ***only for reranking*** to improve recall



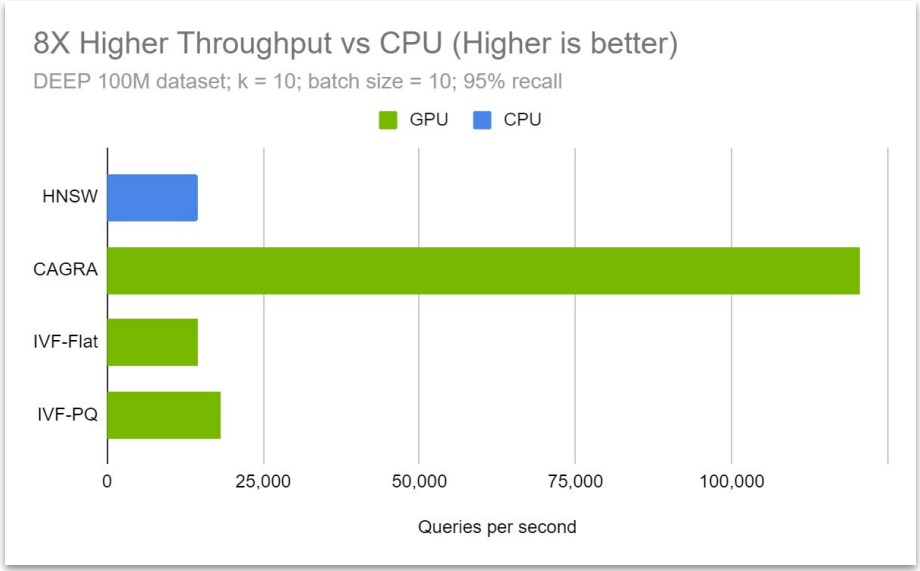
(*) Size in parentheses is the size assumed for DEEP-1B dataset

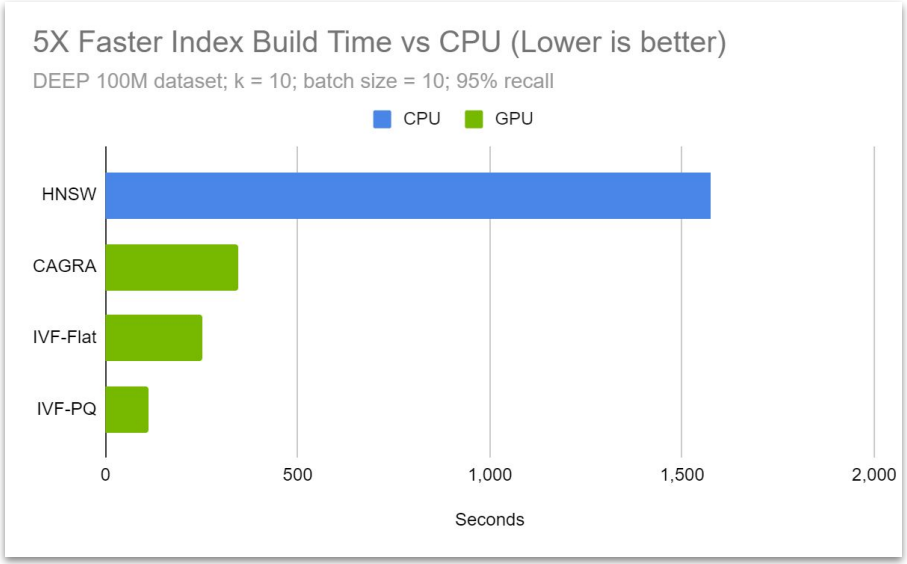


- CAGRA-Q makes a great companion for ***Grace Hopper*** and improved chip-to-chip (C2C) bandwidth.
- **TLDR;** Compressed dataset on device and graph stored in huge page pinned memory has ***equivalent performance*** to original dataset and graph stored on device at high recall levels.

Scaling cuVS - Single GPU (38GB)

Deep 100M; 96 Dimensions





NVIDIA DGX H100 SXM GPU vs Intel Sapphire Rapids 8480CL CPU (224 threads)

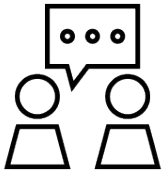
Vector Search Tutorials

Resources

A Variety of Ways to Get Up & Running

More about RAPIDS cuVS

- RAPIDS GTC Talk
- cuVS IVF-PQ GTC Talk
- cuVS CAGRA arXiv
- NVIDIA Tech Blog



Discussion & Support

- Check the RAPIDS cuVS GitHub
- C++ API documentation
- Python API documentation
- Talk to NVIDIA Services

@RAPIDSai

https://github.com/rapidsai/cuvs

https://rapids.ai/slack-invite/

RAPIDS

https://rapids.ai


NVIDIA