SDC | StorageAI™

A SNIA Event

April 29, 2026 • Denver, Colorado

# DiskANN Index Offloading

aka.ms/AboutDiskANN

github.com/Microsoft/DiskANN

Alessandro Goncalves, Solutions Architect, Solidigm
Harsha Simhadri, Partner Researcher, Microsoft

# Agenda

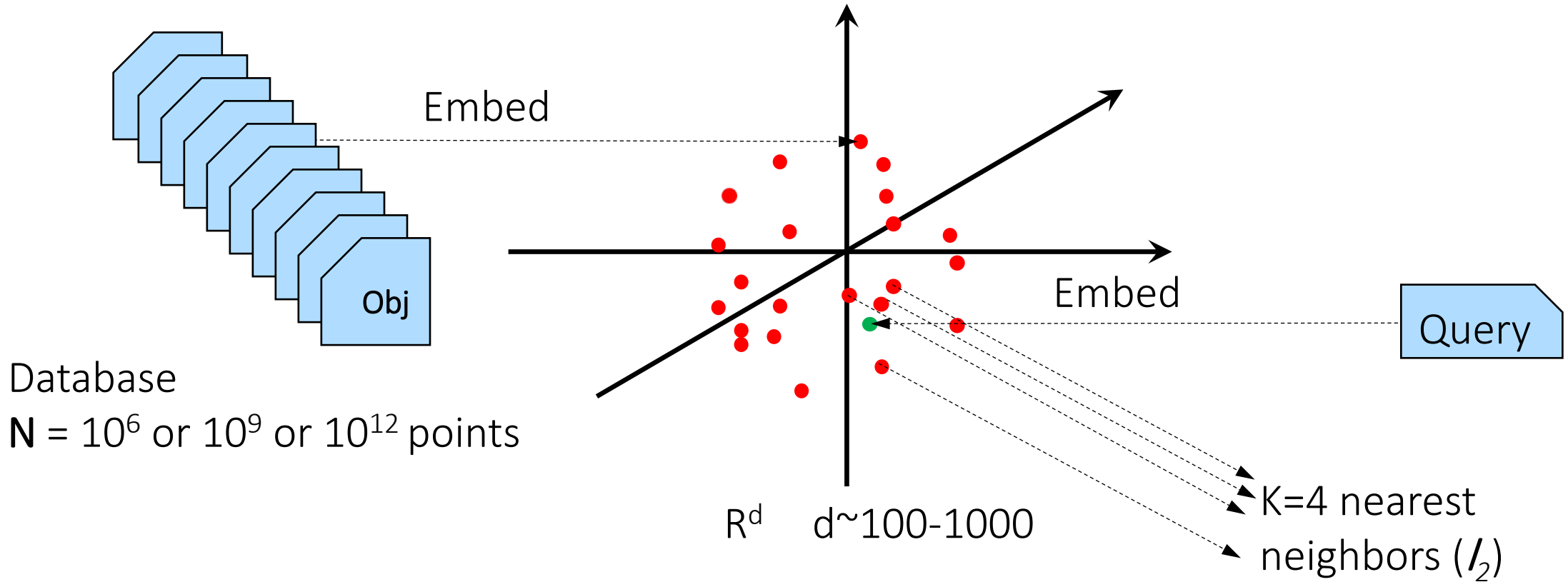
- Approximate Nearest Neighbor/Vector Indices enable Similarity Search
- Vector Search Use Cases
- Vector Indexing Offload Index with DiskANN
- Vector Indexing for any database, memory tier with DiskANNv3

Microsoft

SDC | StorageAI™

# Semantic retrieval with embeddings + ANNS




Exact retrieval might need exhaustive scan, settle for approximate retrieval.
Measure **Recall@k**: fraction of output candidates that are true top-k nearest neighbors

Microsoft

SDC | StorageAI™

# Vector Index Footprint

- Sizing based on:
  - Indexing Technique
  - Number of vectors
  - Dimension
  - Bytes per component (Precision)
    - Float32 (4 bytes) , float16 (2 bytes), int8 (1byte)
- Compare different indexes at same recall level and size.

Microsoft

SDC | StorageAI™

# Vector Search at Microsoft

| @Microsoft | Cloud | | | Edge |
|---|---|---|---|---|
| | **Web Search, Ads & Recommendations** | **Enterprise Email Search** | **Enterprise Doc search** | **Windows Copilot Runtime** |
| **Index Size** | 100s of billions of pages | Millions of indices $10^{14}$+ sentences | Thousands of indices Trillions of paragraphs | ~$10^6$ files, images, text |
| **Update Rate** | ~ billion /day real-time updates | new email, clean up deleted emails | ~1% change/day | As new content as created |
| **Search latency and throughput** | ~10ms latency $10^4$-$10^5$ queries/sec | <100ms, <100 queries/day | <100ms, 100 queries/sec | <100ms, <100 queries/day |

*Very large range of sizes, ingest and query throughput.*

Microsoft

SDC | StorageAI™

# DiskANNv1 – Indexing Billion+ vectors on an SSD.

- DiskANNv1 (like HNSW) builds a graph-based index, one graph node per vector indexed.
  - Vectors are large (1B*400+Byte/vector) => Offload to SSD
  - Graph is large (1B*100 degree*4bytes) => Offload to SSD.
  - Quantized vectors for navigating graph (30-100GB) => Store in memory
- Optimized Disk Layout
  - Offline reordering
  - Graph connections stored alongside vectors for fewer IOs.

Microsoft

SDC | StorageAI™

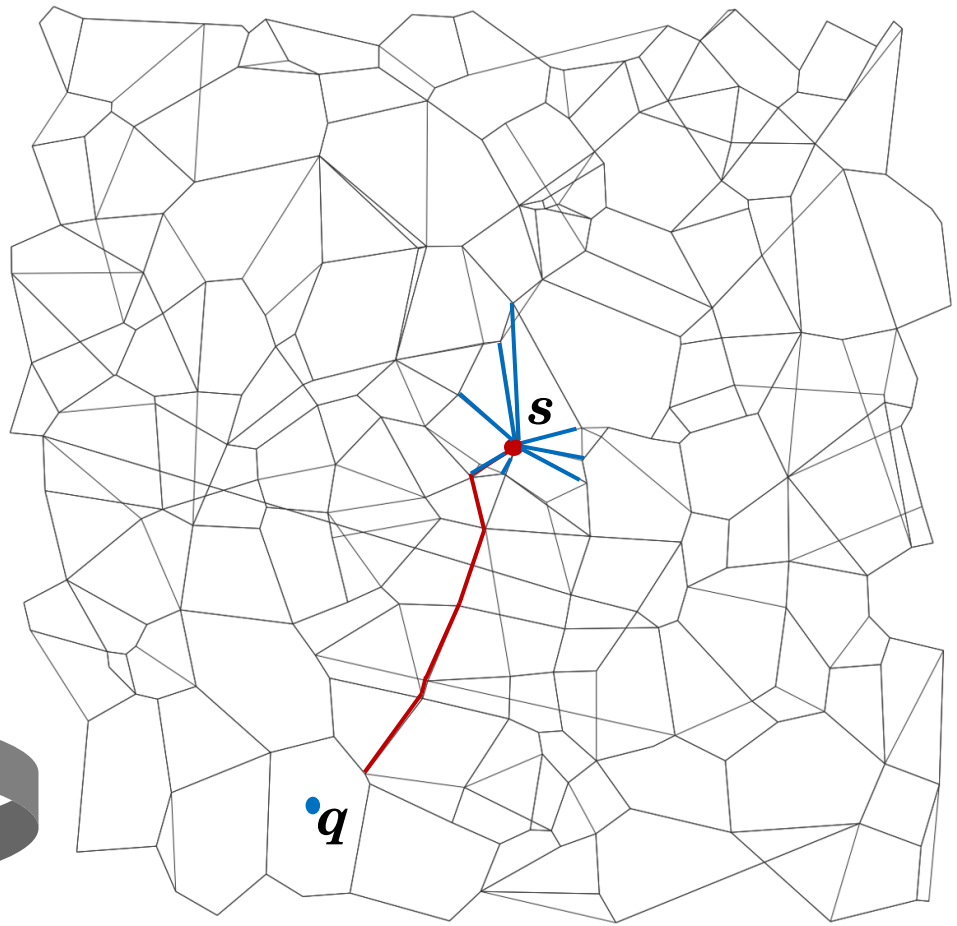
# DiskANN: High recall, low latency via hybrid DRAM+SSD index

[Subramanya, Devvrit, Kadekodi, Krishnaswamy, Simhadri'19]

DRAM: Compressed vectors (~32B)

SSD: 1B Vectors (100-1000d) + Graph(~100 degree) ~ 500GB-1TB
**Low Diameter (<10 hops)**

GreedySearch(*q*)
- Let *p* := *s* (start node)
- Fetch neighbors of p from SSD
- Use **compressed representation of points** to find neighbor *p* closest to *q*

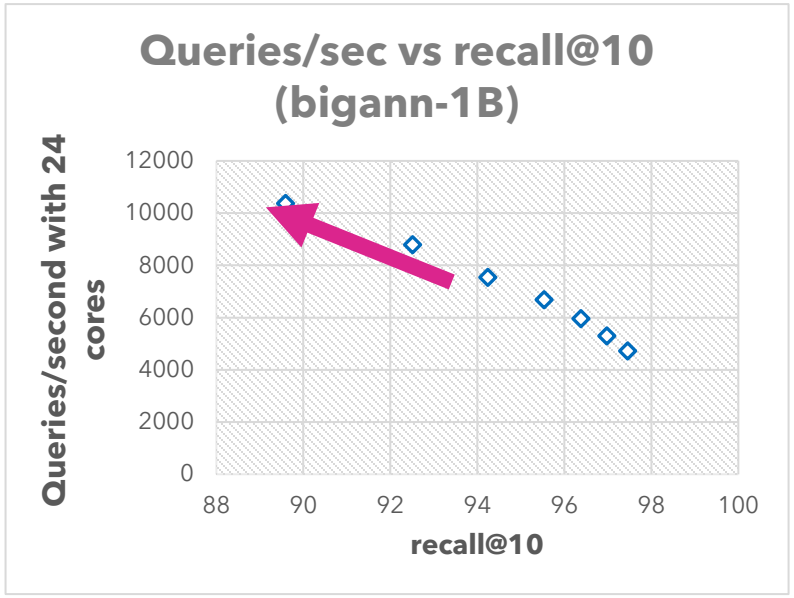


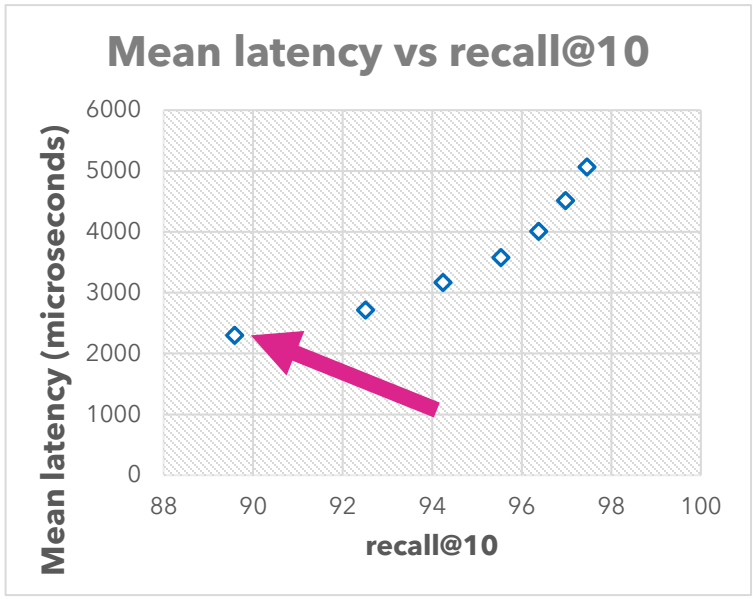

 

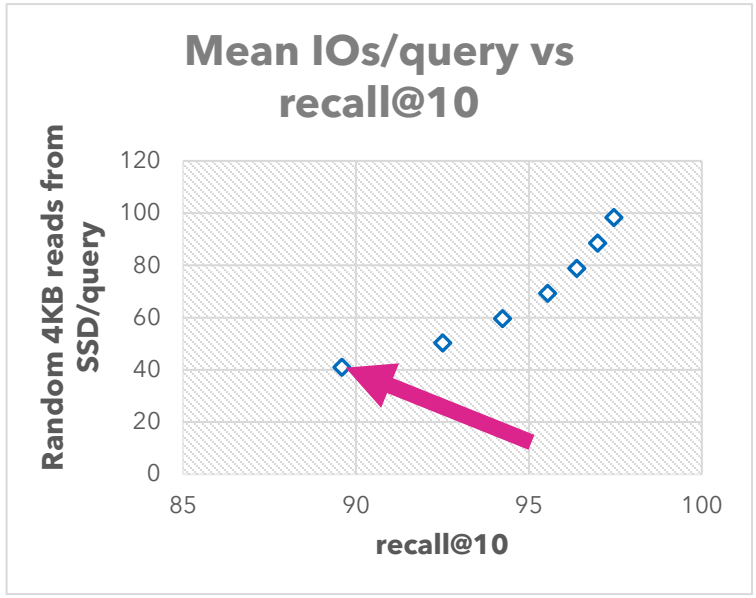
Microsoft

SDC | StorageAI™

# Recall, latency, QPS and IO/s for 1 billion vectors




DiskANN reduced #random 4KB reads needed for 90% recall from 1000s to just 40.

Memory footprint = 32GB + 250K adjacency lists cached in memory ~ 33GB

In comparison, in-memory graph indices (e.g. HNSW) would need 500GB+ DRAM



Microsoft



SDC StorageAI™

# Algorithmic Innovations to enable ANNS at scale

Problem 1: Serving O(100B) indices with <10ms latency and high throughput requires in-mem indices requires 10,000s of machines with O(100GB) DRAM

→ DiskANN [NeurIPS'19]:

Index ~1B vec/machine using SSDs; 5-10x higher density than in-mem; <10ms query latency, 10000+ QPS.

Problem 2: Hard to update, deletes with stable recall is hard. Rebuilt from scratch periodically. 10,000s of machines to periodically rebuild indices every 6/12/24 hours.

→ Fresh-DiskANN[arXiv:2105.09613]
IP-DiskANN[arXiv:2502.13826]

DiskANN + Real-time freshness + 1000s updates/sec/node

Problem 3: Not designed for filtered/hybrid queries. Low recall and high query complexity.

```
e.g., ORDER by L2 distance
      AND (date > Aug 2023)
      AND (brand = x OR y OR z)
```

→ Filtered-DiskANN [WWW'23]:

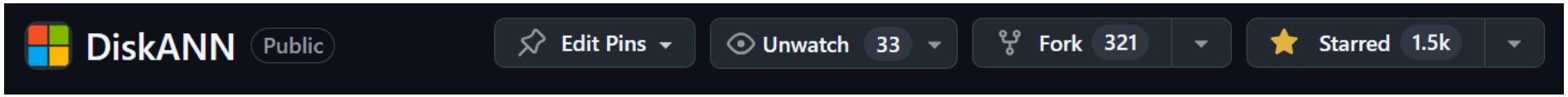
Order(s) of magnitude higher QPS and lower query latency;
High recall for rare predicates.

Microsoft

SDC | StorageAI™

# OSS Release (~2023) and Industry Impact




**Microsoft**

- Bing web search, real-time index
- Advertisements
- Microsoft 365
- Windows

Relevance lifts, new scenarios enabled, and hardware saved

**Adoption in other databases**

- Postgres-> TimescaleDB/Pgvectorscale
- Cassandra ->IBM/Datastax
- SQLite -> Turso
- Milvus and other vector DBs.

**Hardware/Accelerator**

- Kioxia AiSAQ for storage-only solution
- Intel OptaNNE for Optane pmem
- NVIDIA cuVS library

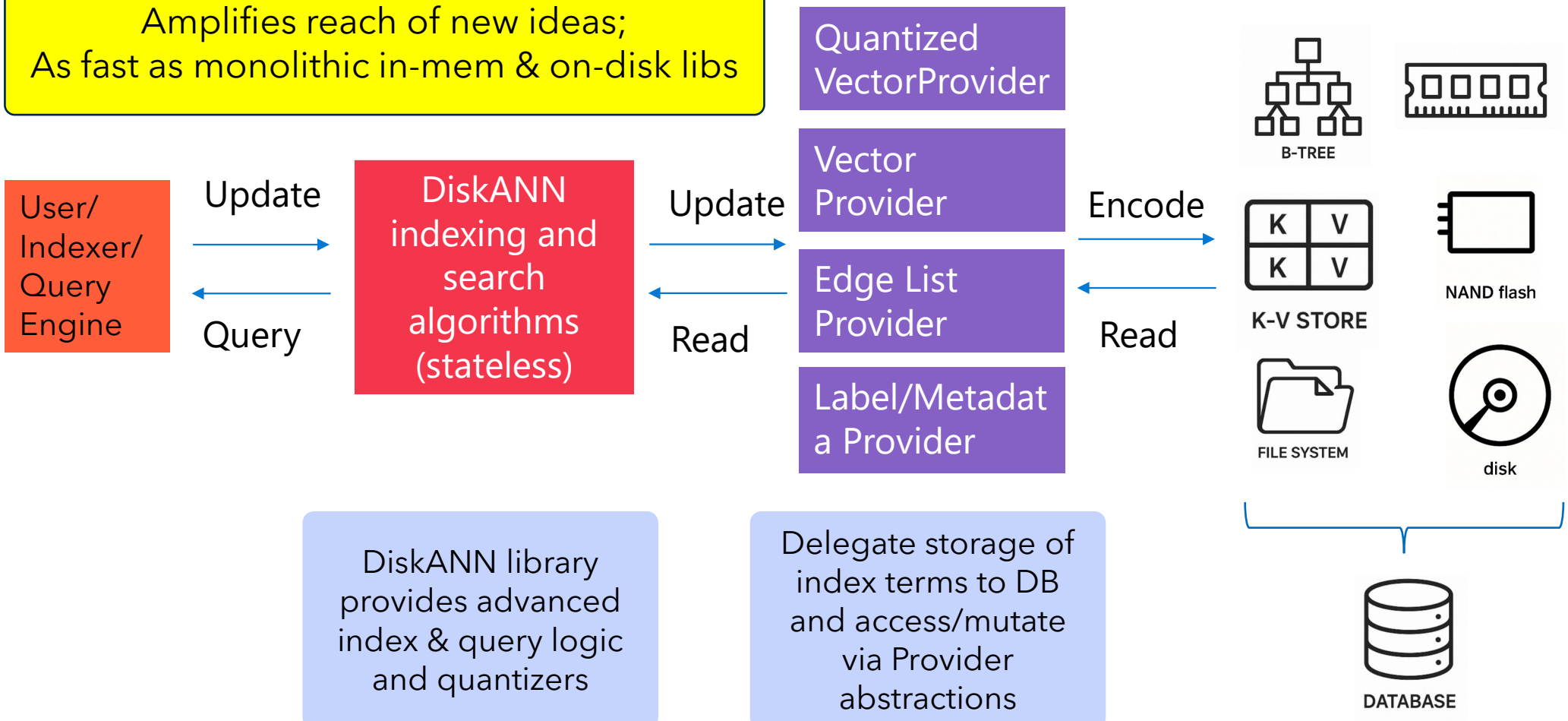
 

Microsoft

SDC | StorageAI™

# DiskANNv3
# In-process stateless library for any DB

One branch for Research and Production;
Amplifies reach of new ideas;
As fast as monolithic in-mem & on-disk libs




Operate at multiple storage tiers

DiskANN library provides advanced index & query logic and quantizers

Delegate storage of index terms to DB and access/mutate via Provider abstractions

Inherit elasticity, availability and durability of host DB

Microsoft

SDC | StorageAI™

# New Algorithmic Challenges

- One Graph – incrementally updated, no merges, no rebuilds
  - [Xu, Manohar, Bernstein, Chandramouli, Wen, Simhadri'25]
- Minibatch updates for parallel, deterministic single-writer update
  - [Manohar, Shen, Blelloch, Dhulipala, Gu, Simhadri, Sun'24]
- Ingest in quantized space for low cost and high performance
- Faster algorithms for vectors + arbitrary predicates that work closely with the database's query planner

Microsoft

SDC | StorageAI™

# DiskANNv3 enables best in class performance

- DiskANNv3 + Garnet k-v *v.* Provisioned Vector DBs
- DiskANNv3 + Cosmos DB *v.* Serverless Vector DBs
- DiskANNv3 + in-mem *v.* hnwslib
- DiskANNv3 + disk *v.* monolithic DiskANN

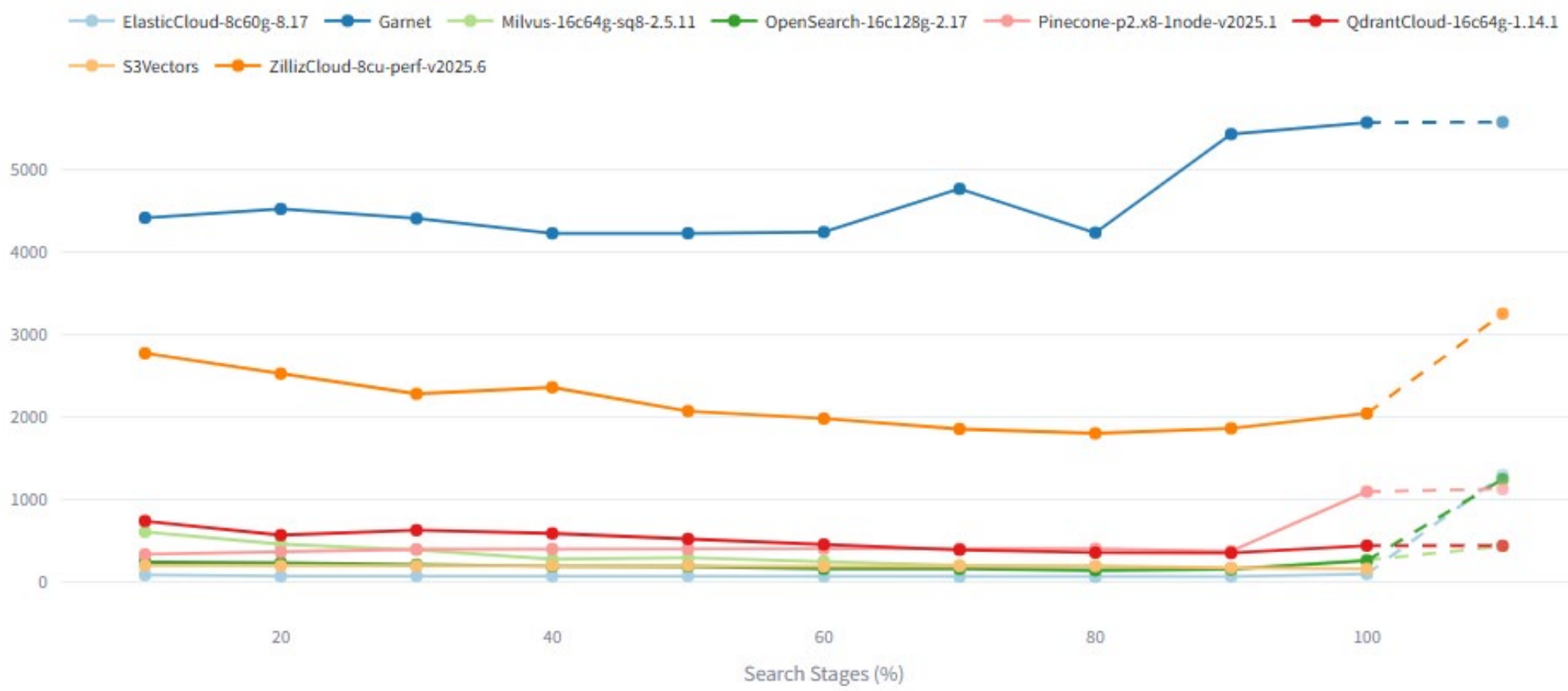
Microsoft

SDC | StorageAI™

# DiskANNv3 + Garnet k-v store v. Provisioned vector databases

Performance with continuous 1000/sec vectors ingested (10M vectors, 768 dimensions)

**2x higher search throughput than Zilliz, 5-10x over others**



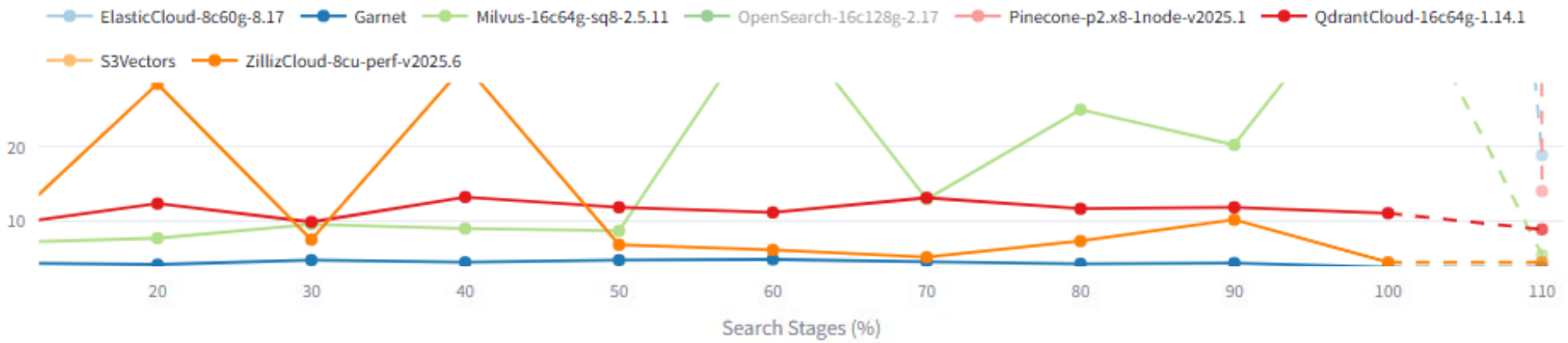



ageAI™

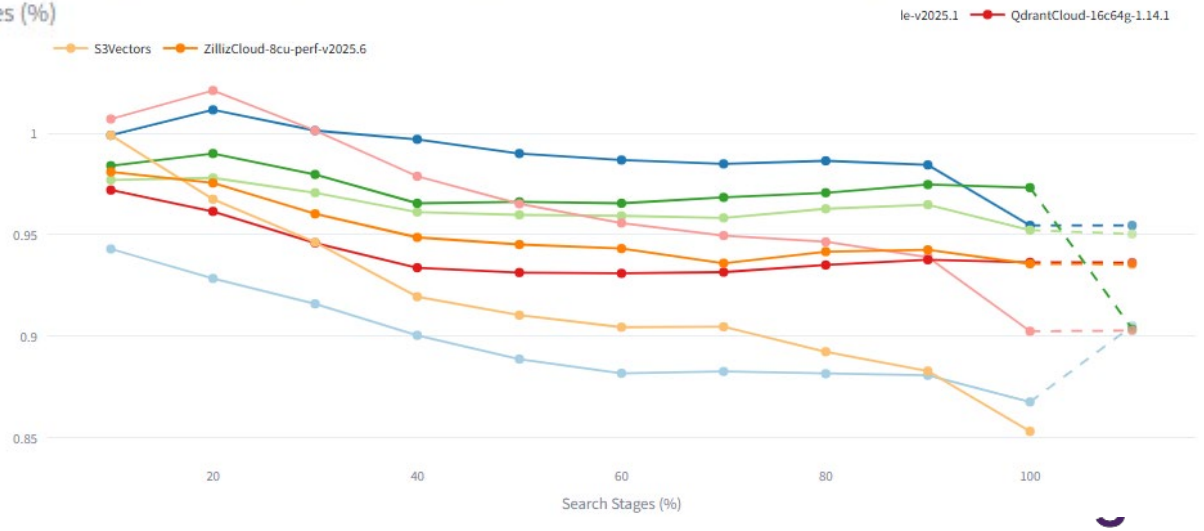
# DiskANNv3 + Garnet outperforms Provisioned vector DBs

Performance with continuous 1000/sec vectors ingested (10M vectors, 768 dimensions)

## Predictable low p99 latency ~ 3ms




## Consistent high recall > 95%@100



Microsoft

# DiskANNv3 + CosmosDB NoSQL => Serverless Vector DB

[aka.ms/CosmosDB/VectorSearch; VLDB'25]

## Performance

- **<20ms query latency on 10M vectors.**
- **< 2X increase in latency/cost as index scales from 100D->768D, 100K->10M vectors.**

## Stable Recall and Latency

- **Stable Recall over long stream of updates even with distribution drift.**
- **No ingestion or query latency spikes due to rebuilds or segment merges.**

## Filtering and Multi-tenancy

- **Filtered queries with upto 8X better P99 cost and latency with new algorithms.**
- **Native Multi-Tenancy: Fast Search and High Recall on a long tail of tenants.**

## Cost and Scale

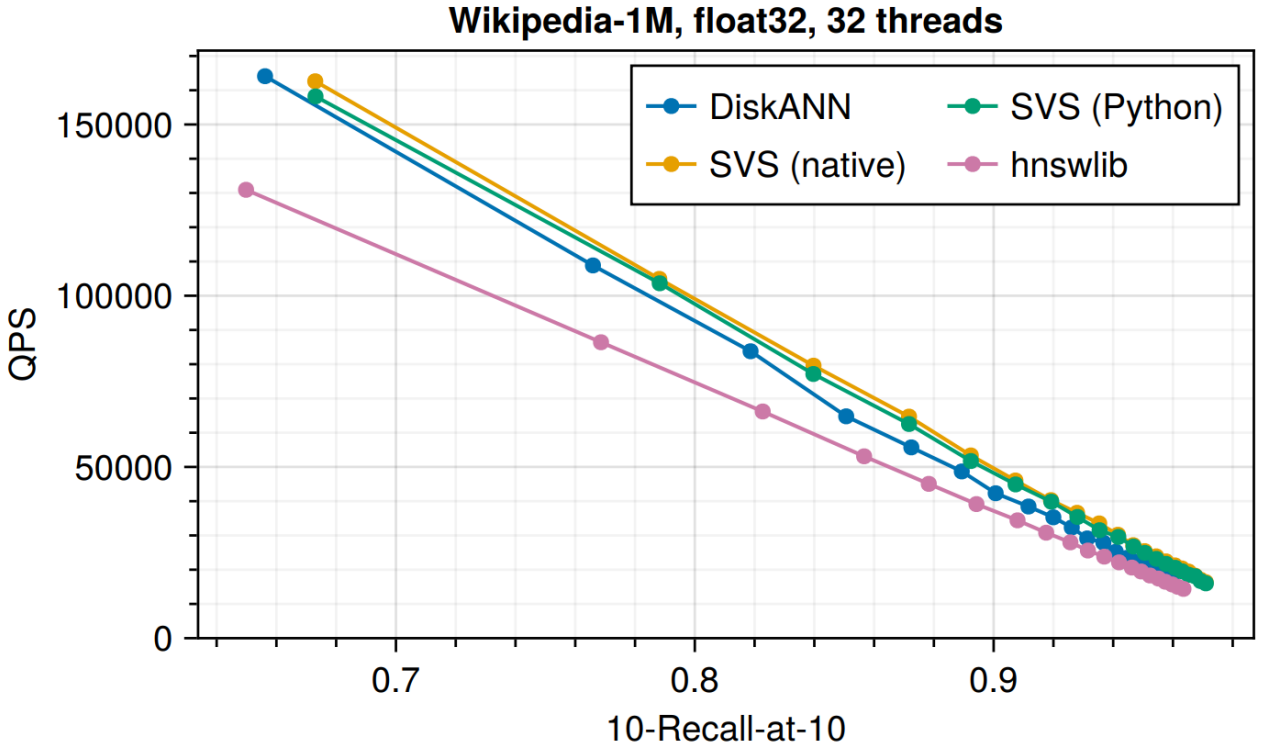
- **Scale out to 1 Billion Vectors with automatic partitioning.**
- **Lower query costs. Upto 43X and 12X than Pinecone and Zilliz respectively at 10M scale**

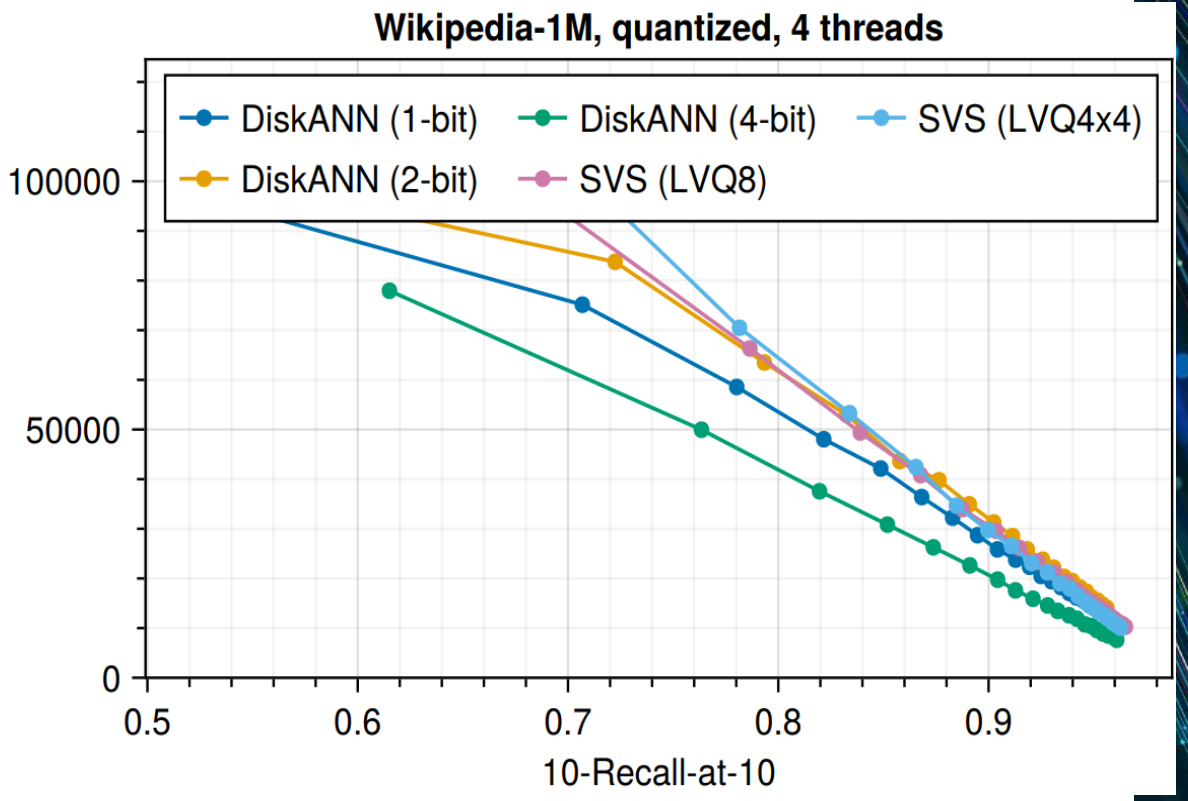
 

Microsoft

SDC | StorageAI™

# DiskANN3 + memory buffers outperforms hnswlib



Microsoft

SDC | StorageAI™

# Conclusion

- Vector indices can be offloaded to storage devices
  - Enables Cost-efficient deployment at scale.
  - Used in Web search, Grounding indices, Ads systems, doc/email search, Windows, Copilots, Azure Databases
- Vector indexing libraries can be designed to composed with many DBs and storage media
  - abstract logic away from storage details to allow instantiation cross N algorithms X M stores/DBs
  - Think bringing vector indexing to your DB or platform rather than vector DBs
- Platform for research (see aka.ms/AboutDiskANN for research papers)
  - Exploring multi-vectors, new distance functions, quantizers and update methods
- OSS: github.com/Microsoft/DiskANN
- The datasets and baselines we released for community are at https://github.com/harsha-simhadri/big-ann-benchmarks
- Let's talk if you want to deploy for your storage device

Microsoft

SDC | StorageAI™


SDC | StorageAI™
A SNIA Event
Thank You
github.com/Microsoft/DiskANN