Closer than you think

Ceph’s journey to nearest neighbor search

Ceph Day Raleigh ‘26

Kyle Bader
Chief Architect, Data and AI, Ceph
IBM Storage

Agenda

What Are Vectors & Vector Search?

What is a vector?

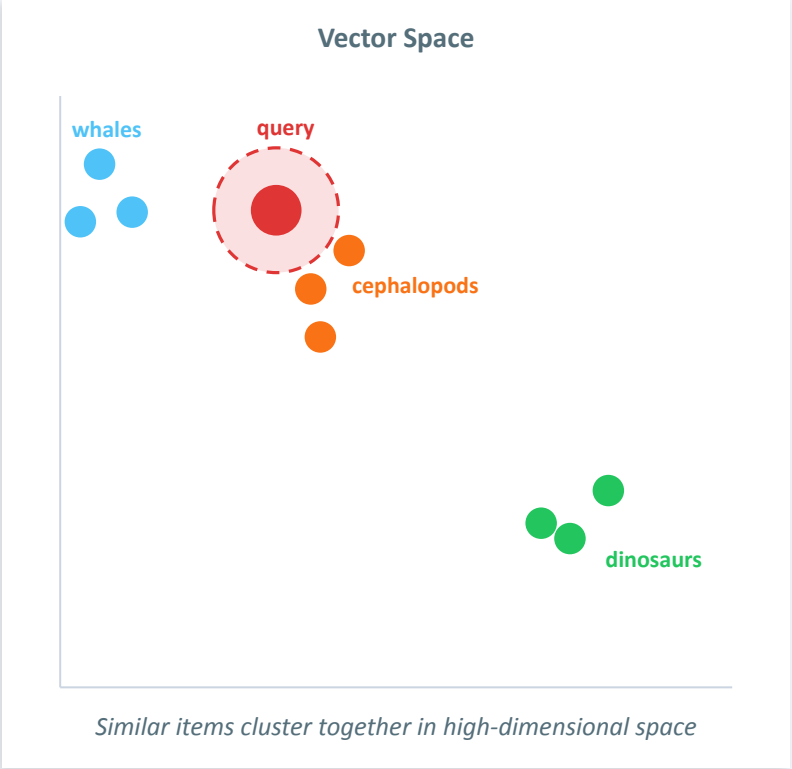
An array of float32 numbers encoding the semantic meaning of any object — text, image, audio — as a point in high-dimensional space.

How are they created?

Embedding models (OpenAI, Cohere, sentence-transformers) map raw data to fixed-length numeric arrays, placing semantically similar items close together.

Approximate Nearest Neighbor (ANN)

Finds the K most semantically similar items to a query vector — the engine behind semantic search, recommendations, and RAG pipelines.



Similar items cluster together in high-dimensional space

What is S3 Vectors?

AWS's extension to S3 that adds native vector storage and similarity search — no separate vector database required.

Vector Bucket

New bucket type. Only s3vectors:* actions are permitted. Supports SSE-S3 and SSE-KMS. Internally maps to a LanceDB dataset directory.

Index

A LanceDB table inside a bucket. Bound to a distance metric and dimension count at creation time via CreateIndex.

Vectors

Dense float32 arrays stored with string metadata tags. Up to 10 K/V pairs, 2 KB filterable, 40 KB total per vector.

Query

Top-K ANN search via QueryVectors. Optional metadata tag predicate narrows the candidate set before the ANN phase.

Bucket Directory Structure (hidden)

```
s3://foo/            ← Vector Bucket
├── bar/              ← Index 'bar'
│   ├── data/
│   ├── indices/
│   ├── _latest.manifest
│   └── schema.arrow
└── baz/              ← Index 'baz'
```

ARN Formats

```
arn:aws:s3vectors:region:ACCT:bucket/my-bucket

arn:aws:s3vectors:region:ACCT:bucket/my-bucket/index/my-index
```

Our Goals

01

Cost-Effective Storage

Designed for billions of vectors. Target ≤1:3 ratio of raw vector data to total on-disk footprint (data + IVF-PQ indexes). Leverages Ceph's erasure coding and tiering.

02

Zero Extra Infrastructure

No new services to deploy or monitor. Vector capabilities live entirely inside the RGW request pipeline — if you run Ceph, you get vector search automatically.

03

Sub-Second ANN Search

Target query latency of 100–800 ms. Achieved via IVF-PQ on-disk indexes with tunable nprobes — trading recall precision for latency as needed.

Design Parameters

Parameter	Value	Notes
Indexes per vector bucket	≤ 10,000	*Each index independently queryable with its own metric and dimensions*
Vector dimensions	1 - 4,096	*Dense float32 only; sparse vectors are out of scope for v1*
Data type	float32	*Single-precision IEEE 754 — matches native embedding model output*
Top-K per query	≤ 30	*Maximum nearest neighbors returned per QueryVectors call*
Space amplification	≤ 1 : 3	*Raw vector data to total on-disk footprint including IVF-PQ index files*
PutVectors batch size	500 / 20MB	*Per call limit — whichever is smaller; max 5 calls/index/second*

Adding S3 Vectors to Ceph

A thin layer over RGW — heavy lifting delegated to LanceDB, persisted through the existing RADOS storage abstraction.

1

New Resources & IAM Actions

- Vector bucket type
- Index resource type
- Vector CRUD ops
- IAM policy actions
- S3-compatible routing
- SSE-S3 / SSE-KMS

2

RGW Request Handling

- Parse & validate
- Enforce dimensions
- Check batch limits
- Rate limiting
- Dispatch to engine

3

LanceDB (Rust via FFI)

- Build IVF-PQ indexes
- Cosine & L2 distance
- merge_insert writes
- Background optimize
- Metadata filtering

4

RGW Storage Abstraction (SAL)

- Persist Lance datasets
- Inherit replication
- Leverage tiering & EC
- No new daemons
- Existing auth & quotas

Library Assessment: FAISS · DiskANN · LanceDB

Three credible ANN libraries evaluated against our requirements — on-disk operation, distance metrics, integration effort, and licensing.

FAISS

DISQUALIFIED

Meta Research · MIT License

Distance metrics

L2, IP, Cosine

Storage model

In-memory first

Language

C++ / Python

GPU acceleration

Yes (CUDA native)

On-disk ANN

❌ Not primary use case

Integration effort

High — no built-in I/O

Verdict

Optimized for GPU RAM, not disk. Out of scope.

DiskANN

VIABLE

Microsoft · MIT License

Distance metrics

L2, Cosine, Inner Product

Storage model

Disk-native (SSD-optimized)

Language

C++

GPU acceleration

Yes (cuVS)

On-disk ANN

✓ Core design

Integration effort

High — no query engine, format, or I/O layer

Verdict

Good algorithm, but needs substantial glue code.

LanceDB

SELECTED

LanceDB Inc. · Apache 2.0

Distance metrics

L2, Cosine

Storage model

Lance columnar, disk-native

Language

Rust (FFI / C API)

GPU acceleration

PyTorch (cuVS roadmap)

On-disk ANN

✓ IVF-PQ, AVX2 SIMD

Integration effort

Low — full-stack library

Verdict

Best fit: metrics, storage, hybrid search, clean API.

Why LanceDB? — Technical Deep Dive

IVF-PQ Index

Composite of Inverted File Index (IVF) and Product Quantization (PQ). IVF divides the vector space into cells; PQ compresses vectors to reduce index size. Tunable nprobes per query — higher values increase recall, lower values reduce latency. Target: 100–800 ms.

SIMD Acceleration (AVX2)

K-means clustering during index generation is accelerated via AVX2 SIMD. Minimally requires AVX2 — relevant for modern x86 Ceph OSD nodes.

merge_insert for PutVectors

Atomic upsert — checks existence and inserts or updates in a single call. Avoids race conditions that would arise from separate read-then-write. A background optimize() call rebuilds the IVF-PQ index without blocking concurrent merge_insert or query calls.

GPU Acceleration Path

Index generation can be GPU-accelerated via PyTorch (accelerator='cuda'). Ceph gateway nodes typically lack accelerators, but this could change. Might need to look into native approach with cuVS, though.

FFI / C API: LanceDB is Rust-based. Integration into Ceph's C++ codebase via Foreign Function Interfaces or a dedicated C API wrapper.

Metadata Filtering Strategy

S3 Vectors supports per-vector metadata tags — filterable and non-filterable — with up to 10 K/V pairs per vector.

The Challenge

Different vectors within the same index can have different metadata keys. Flattening every key into its own column would produce an unbounded number of columns — impractical to pre-create.

LanceDB Scalar Index Types

- btree
- bitmap
- label_list (equality only after flattening, no range queries)

Frequency-based flattening

An accumulator in the PutVectors path tracks the most frequently seen filterable keys per index. Top-N keys are dynamically flattened into dedicated columns with scalar indexes.

Query-driven indexing

Monitor which keys are most frequently used as filters in QueryVectors calls. Proactively create scalar indexes on hot keys — prioritizes latency for real-world query patterns.

Post-filtering

Inflate top-k request size for ANN search and apply JSON filters to the result, returning top-k survivors. Maintains compatibility with S3 Vectors. Extension to provide filterableMetadataKeys during CreateIndex for optimized pre-filtering.

When to Use OpenSearch Instead?

✓ Reach for S3 Vectors when…	→ Reach for OpenSearch / Dedicated VDB when…
You already run Ceph, data already in Ceph	You operate a dedicated search platform
Billions of vectors, cost is the priority	Single-digit ms latency SLAs required
Object storage + vectors in one system	Rich full-text + vector hybrid search
Simple ANN with optional tag filtering	Complex query DSL, faceted or range search
No new infrastructure budget or ops headroom	Dedicated ops team and infrastructure budget

S3 Vectors is not a replacement for OpenSearch — it is a cost-effective addition for workloads already living in the object store.

Progress So Far

Three open pull requests that together form the foundation of Ceph's S3 Vectors implementation.

`yuvalif/lancedb-c` · `PR #3`

MERGED

Background Processes & C Bindings

- S3 Vectors simulator over the Rust-C FFI layer — exercises the full create/put/query/delete cycle
- Concurrent index rebuild: background optimize() triggered when unindexed row threshold is exceeded, non-blocking to put/query
- lancedb_table_index_stats() FFI — replaced external JSON counters with LanceDB's native manifest stats
- Dynamic schema system: scalar columns defined at CreateIndex time, typed Arrow builders per column
- Test dataset: 693 Ceph source embeddings (1k-dim) with function/class/line metadata

`ceph/ceph` · `PR #66409`

CHERRY-PICKED → #66066

rgw/s3vector: Add Basic LanceDB Support

- CreateIndex, DeleteIndex, GetIndex, ListIndexes fully implemented
- PutVectors (merge_insert), GetVectors, DeleteVectors, ListVectors implemented
- QueryVectors with IVF-PQ ANN search implemented
- SigV4 auth support; teuthology integration test suite passing on CentOS
- Cherry-picked into the main wip-s3vector branch (#66066)

`ceph/ceph` · `PR #66066`

OPEN · IN REVIEW

rgw/s3vector: S3 Vectors API Support in RGW

- Main integration PR — 34 tracked tasks covering the full API surface
- Vector bucket CRUD (CreateVectorBucket, GetVectorBucket, ListVectorBuckets, DeleteVectorBucket)
- VectorBucket policy APIs (PutVectorBucketPolicy, GetVectorBucketPolicy, DeleteVectorBucketPolicy) — in progress
- Cascade deletion of all indexes on DeleteVectorBucket
- VectorBucket attributes to cache schema & distance metric per index — planned

Demonstration

https://asciinema.org/a/8q6k2beIpOLSHrV9

More stuff

Ceph S3 Vectors blog post:

https://ceph.io/en/news/blog/2026/s3-vectors/

LanceDB:

https://github.com/lancedb/lancedb

DiskANN paper:

https://suhasjs.github.io/files/diskann_neurips19.pdf

FAISS papers:

https://arxiv.org/pdf/2401.08281

https://arxiv.org/pdf/1702.08734