~~Billion~~ Trillion-Scale ~~Vector~~ Search on Object Storage

turbopuffer

turbopuffer

fast search on object storage @ 10M WPS scale

semantic search
vector similarity

full-text search
traditional search by keywords

aggregations & group by
real database queries

object storage (S3 / GCS), with adaptable SSD/RAM caching
cheap and scalable

turbopuffer

who’s puffin’

CURSOR Notion Linear Tolan

Pylon Readwise top 3 ai lab SUPERHUMAN warp

clay TELUS *.. and many more (white text is annoying, ran out of time)*

turbopuffer

if you GMI as a new DB, 2 ingredients needed:

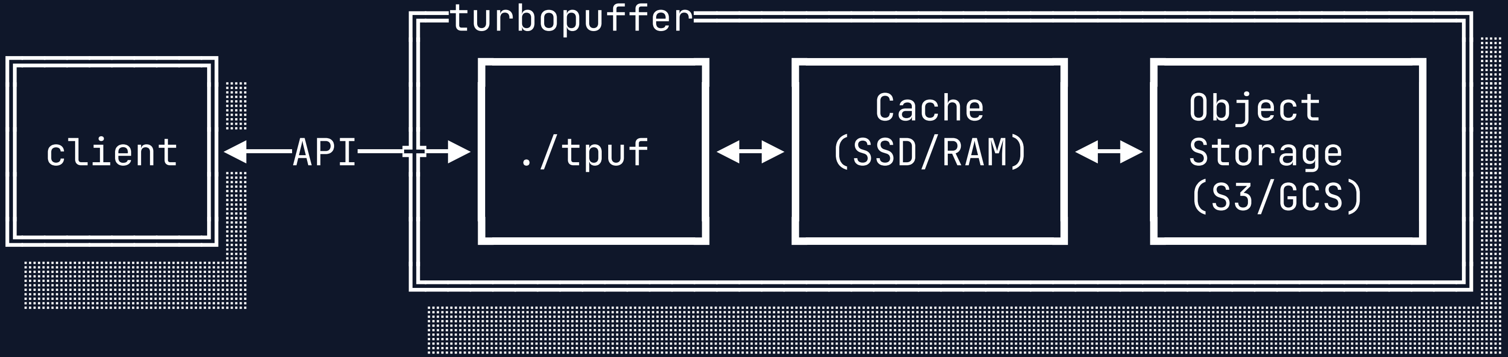
1. **New Workload:** Connect LLMs to data
   Vectors are a 10-30x size amplification
   1kb text → 4x 1024d vectors → ~16kb vectors

2. **New Storage Architecture:** Object-Storage Native
   a. **NVMe SSDs in Cloud (2017)**
   b. **S3 Strongly Consistent (2020)**
   c. **S3 CAS (2024)**

(every successful database will support every SQL query eventually)

turbopuffer

first object storage native database



turbopuffer

economics for object storage native databases

Configuration (storage only)	$/GB (USD)
1 × RAM (100% full)	$5
3 × SSD (50% full)	$0.60
2 × SSD (50% full)	$0.40
1 × SSD cache (100%) + S3†	$0.12
S3†	$0.02

† Storage compute separation

Assuming ~$0.10/GB for NVMe and EBS, ~$0.02/GB for S3

turbopuffer

roundtrip sensitive database w/ high concurrency

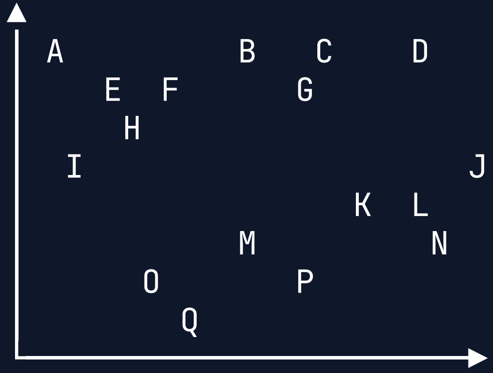
Medium	Random Read	Throughput
RAM	100 ns	25 GB/s
NVMe SSD	100 μs	10 GB/s
EBS†	1 ms	5 GB/s
S3 Hedged p99†	200 ms	5 GB/s

† Compete for network bandwidth (on some machines)

turbopuffer

vector search indexes on object storage

Raw Vectors



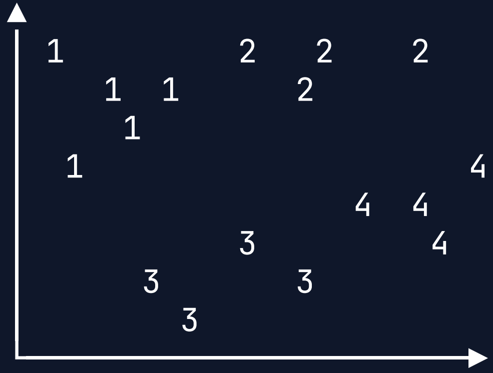
Graph Index

```
(A)-B       D
|   |      /
E---F  C--G      J---K---L
|   |  |         |   |   |
I---H--+----+----+---M---N
|               |
O---P---Q-------+
```

⩽ 6 Roundtrips

Clustered Index

c1, c2, c3, c4



2 Roundtrips

turbopuffer

trade-offs for an object-storage first database

✅ STRENGTHS

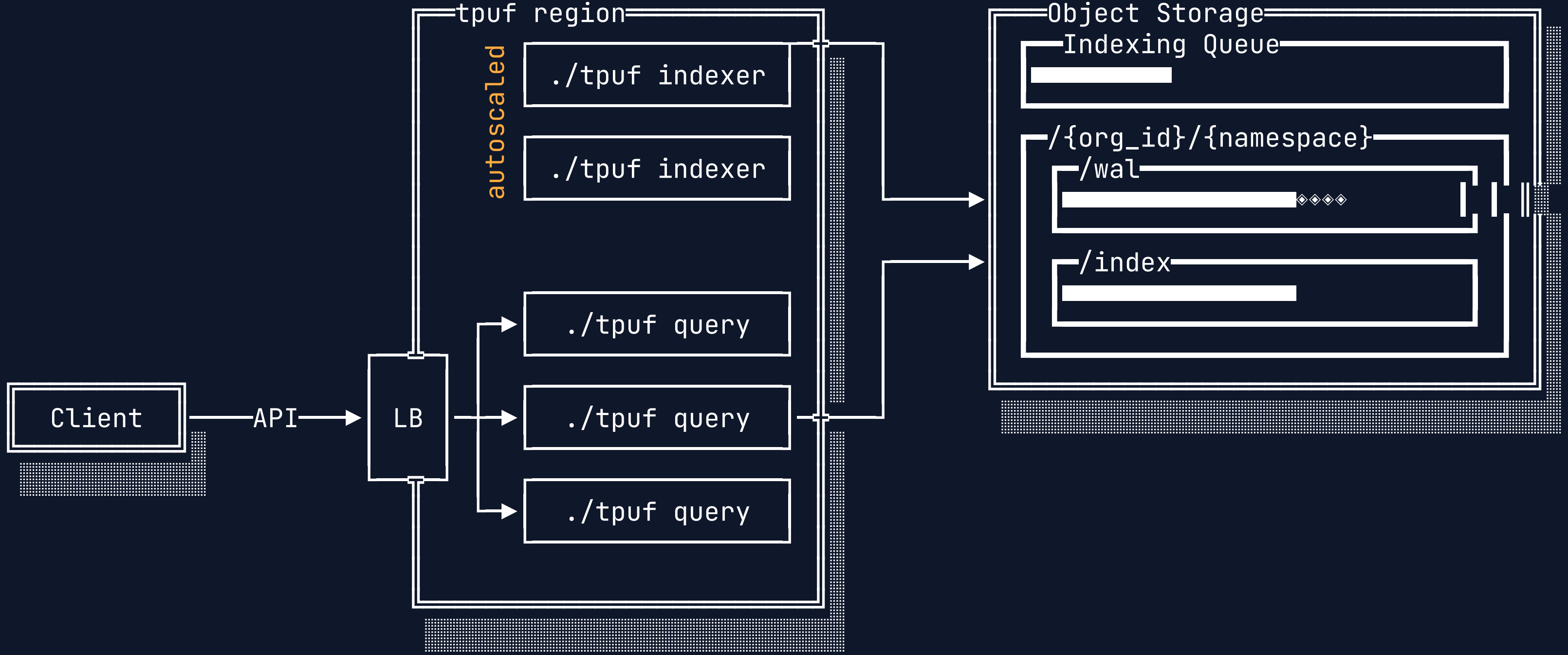
- Low cost
- Simple → reliable and horizontally scalable
- Fast warm queries
- High write throughput

⚠ LIMITATIONS

- Cold queries slow
  ↳ Mitigation: keep full index warm on lower-cost SSDs
- Higher write latency

turbopuffer

architecture you want to be oncall for



turbopuffer

performance

pctile	Warm namespace	Cold namespace
p50	8 ms	343 ms
p90	10 ms	444 ms
p99	35 ms	554 ms

VECTOR SEARCH – 768 dimensions, 1M docs, ~3GB

pctile	Warm namespace	Cold namespace
p50	11 ms	221 ms
p90	18 ms	285 ms
p99	40 ms	433 ms

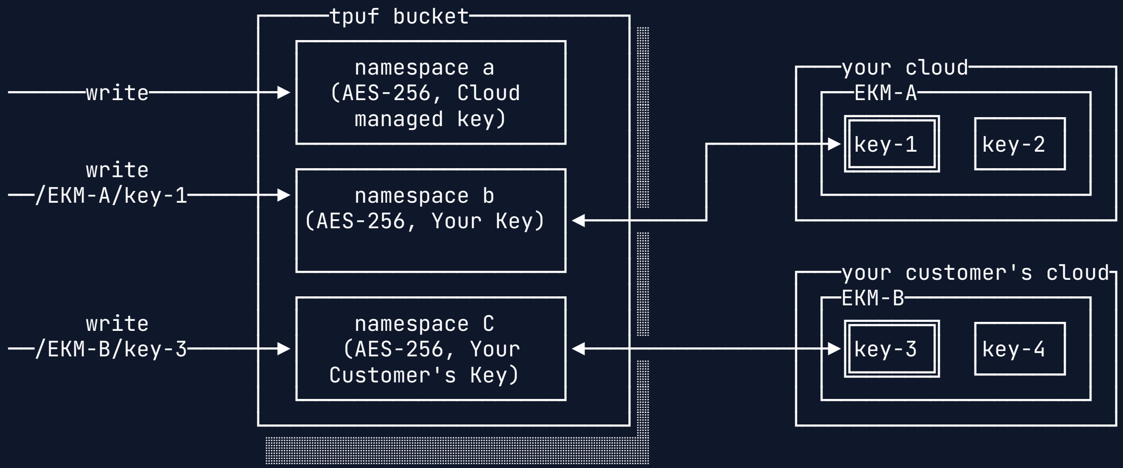
FULL-TEXT SEARCH – BM25, 1M docs, ~300MB

turbopuffer

security

all data at rest is encrypted using AES-256

turbopuffer also supports customer managed encryption keys (CMEK)



CASE STUDIES

CURSOR

> After switching our vector db to @turbopuffer, we're saving an order of magnitude in costs and dealing with far less complexity!
>
> -Aman Sanger, Co-founder

95%
cost reduction

100B+
vectors

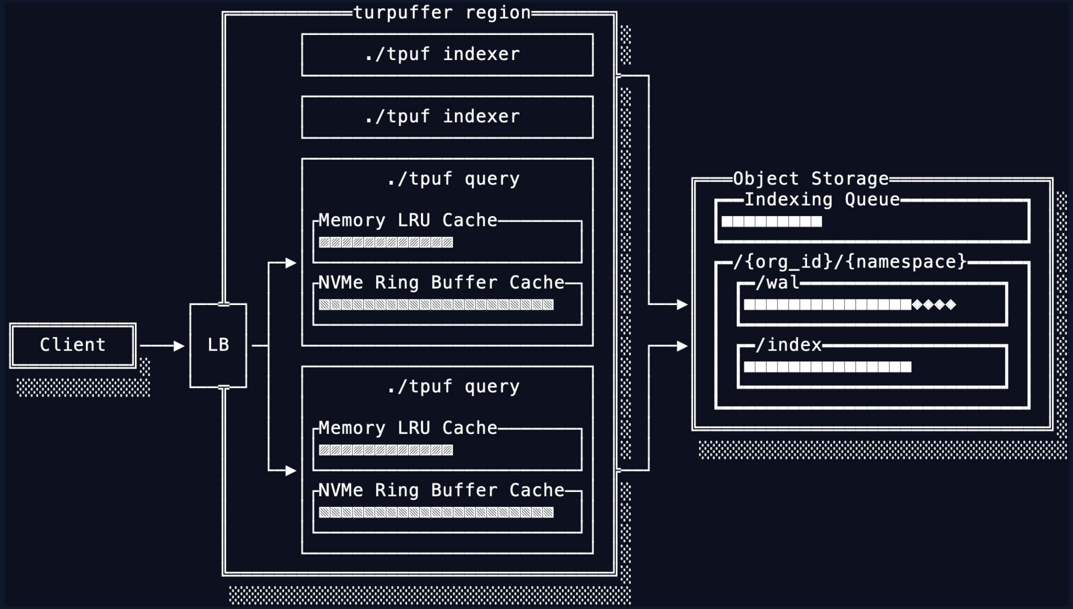
10GB/s
write peaks

10M+
namespaces

* namespace-per-codebase

* active codebase namespaces are loaded into memory/NVMe, inactive fade into object storage

* unlimited namespaces in a fully serverless model; no more bin-packing codebase vector indexes to servers



Notion

> turbopuffer's economics have changed the way we think about building products that connect data to users and LLMs.
>
> -Akshay Kothari, Co-founder

millions
$ saved annually

10B+
vectors

1GB/s
write peaks

1M+
namespaces

* Consistent reads with 100,000+ writes/s peaks

* 80% reduction in cost, allowing Notion to remove per-user AI charges

* ⩾ 99.99% uptime

* From concerned to excited about 10x'ing their data size

* Zero performance drops

* A turbopuffer team so responsive they felt part of the Notion engineering team

* A roadmap aligned with their anticipated needs

Linear

> Their responsiveness and shipping velocity make us feel like we are their only customer.
>
> -Tom Moor, Head of Engineering

70%
cost reduction

250M+
documents

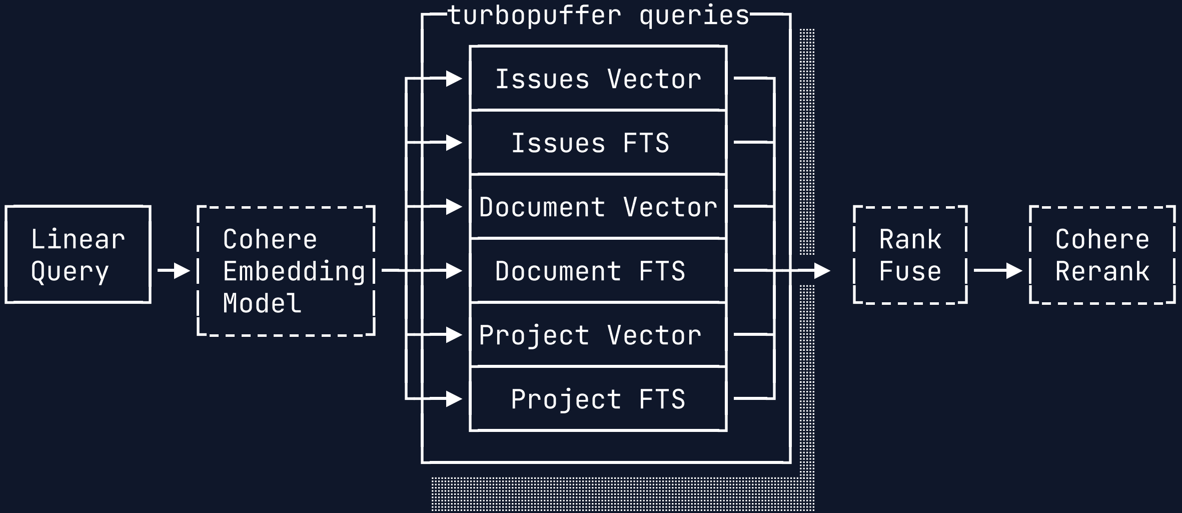
13ms
P50 latency

1.5M+
namespaces

* replaced Elasticsearch & pg_vector

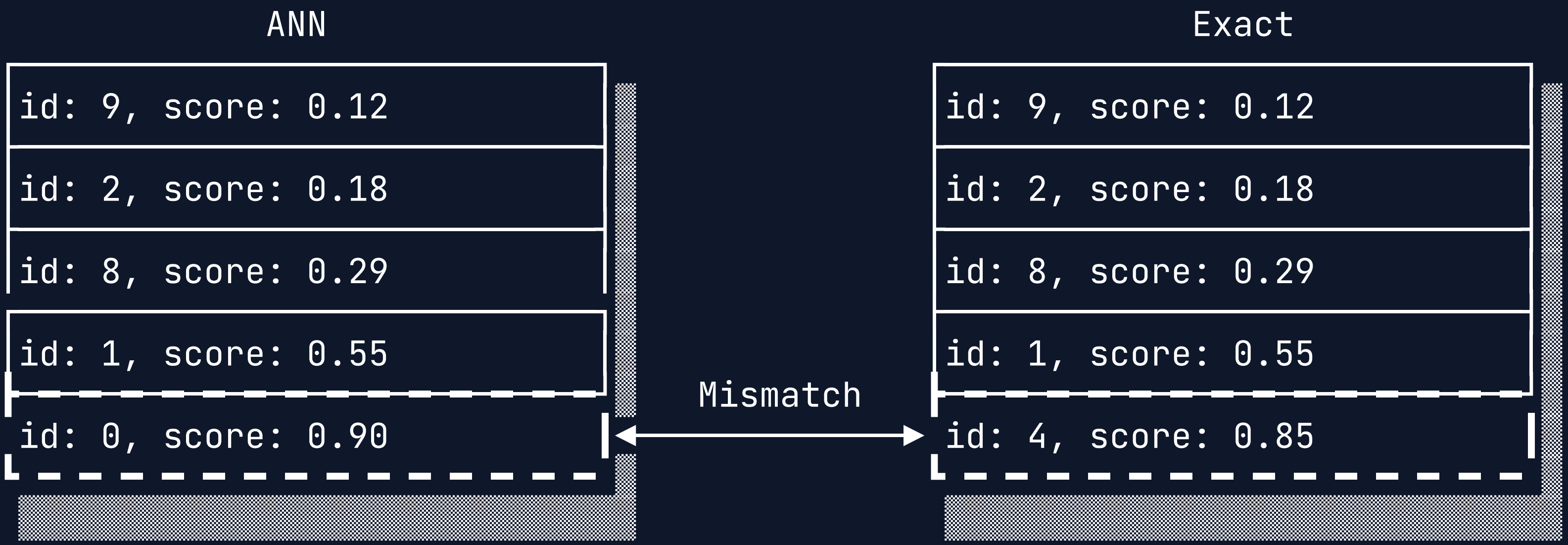
* zero-ops search for terabytes of data

* results from parallel queries, using vector + FTS, passed into a reranker



turbopuffer

recall



ANN

id: 9, score: 0.12

id: 2, score: 0.18

id: 8, score: 0.29

id: 1, score: 0.55

id: 0, score: 0.90

Exact

id: 9, score: 0.12

id: 2, score: 0.18

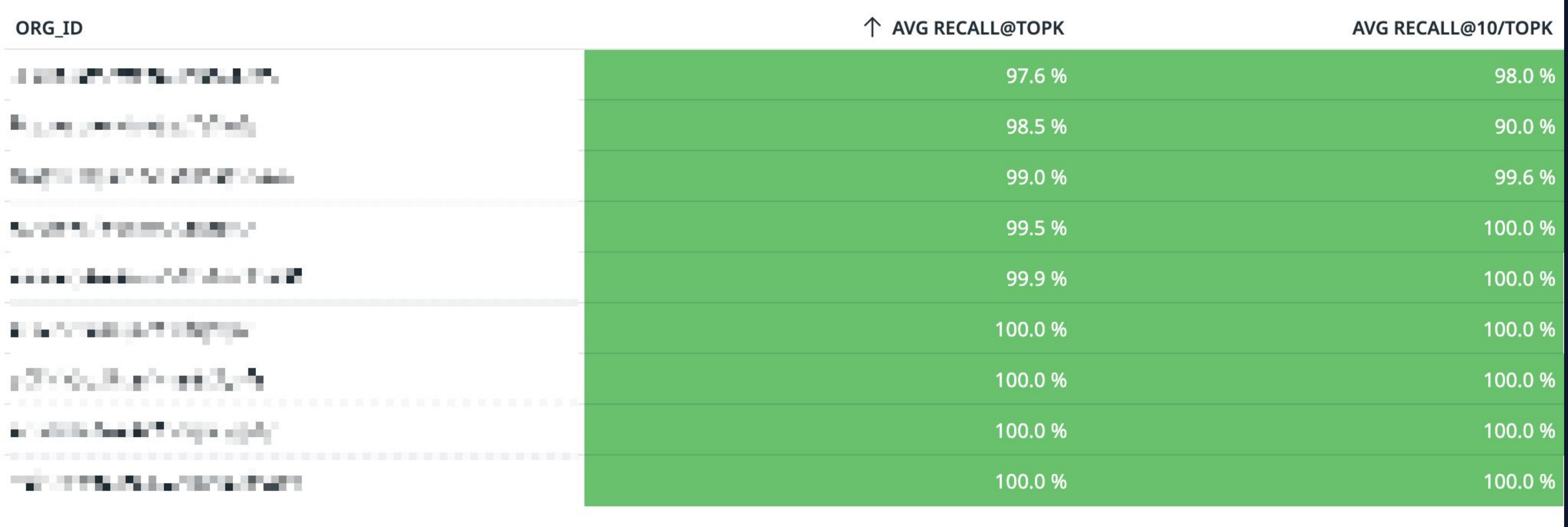
id: 8, score: 0.29

id: 1, score: 0.55

id: 4, score: 0.85

turbopuffer

recall observability



ORG_ID	↑ AVG RECALL@TOPK	AVG RECALL@10/TOPK
[redacted]	97.6 %	98.0 %
[redacted]	98.5 %	90.0 %
[redacted]	99.0 %	99.6 %
[redacted]	99.5 %	100.0 %
[redacted]	99.9 %	100.0 %
[redacted]	100.0 %	100.0 %
[redacted]	100.0 %	100.0 %
[redacted]	100.0 %	100.0 %
[redacted]	100.0 %	100.0 %

turbopuffer

Q&A

APPENDIX

turbopuffer

filtered recall (prefiltering vs postfiltering vs native)

```
|-----------------------------------------------------|
|        -                                            |
|                                                     |
|           -         -                               |
|  -                                                  |
|     -                -                              |
|                 -           -                       |
|         Q       -         -                         |
|                                                     |
|                             -                  +    |
|     -                                               |
|                        -                            |
|                           -                  +      |
|                                            +        |
|                                     +        +      |
|                                            +        |
|                                      +              |
|                                                     |
|-----------------------------------------------------|
```

```
legend:
 Q:  query vector
 +:  document matching the filter
 -:  document not matching the filter
```