



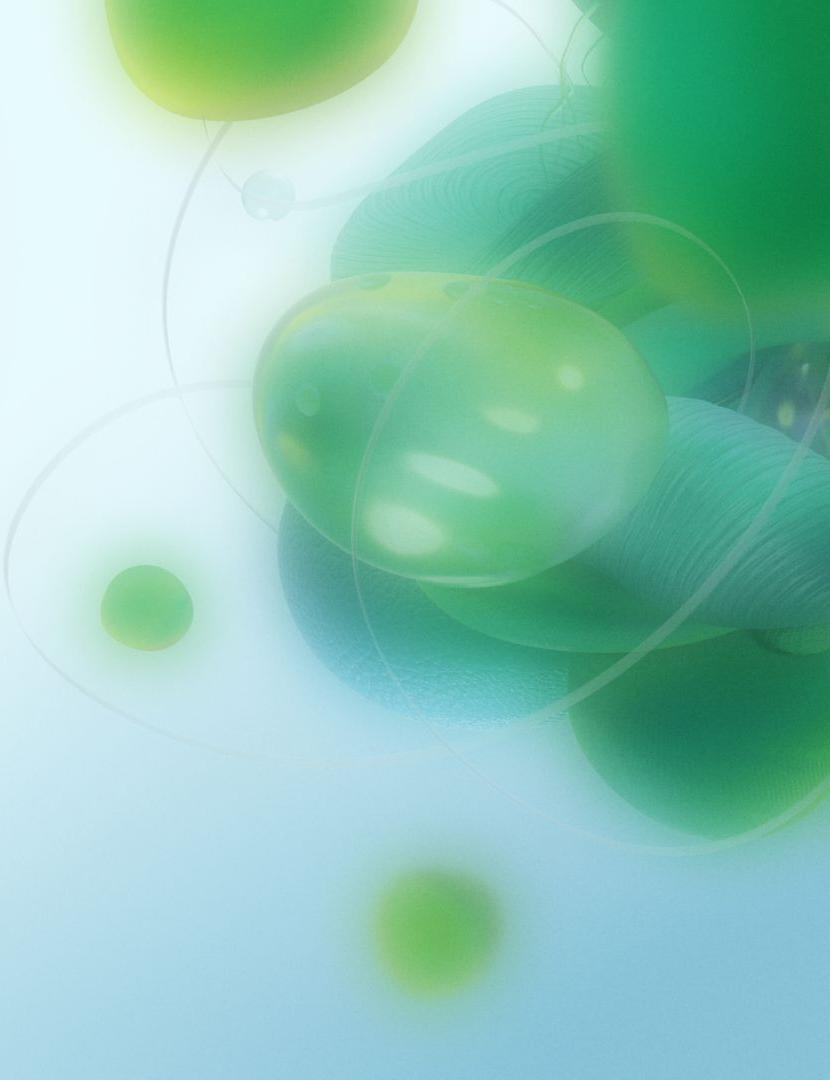
GenAI Beyond Prototyping: The Path to Production with AI-Native Databases

JP Hwang
Developer Educator



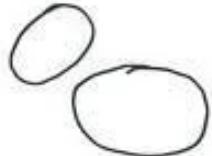
Let's talk about...

**Going to
production**

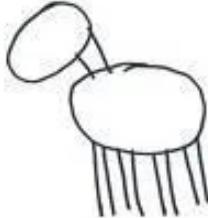


HOW TO:
DRAW A HORSE

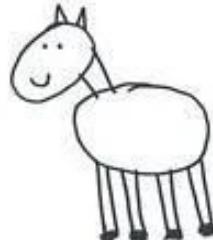
BY VAN OKTOP



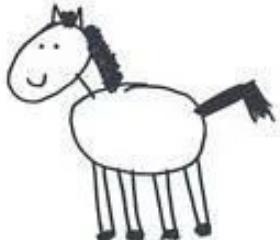
① DRAW 2 CIRCLE



② DRAW THE LEGS

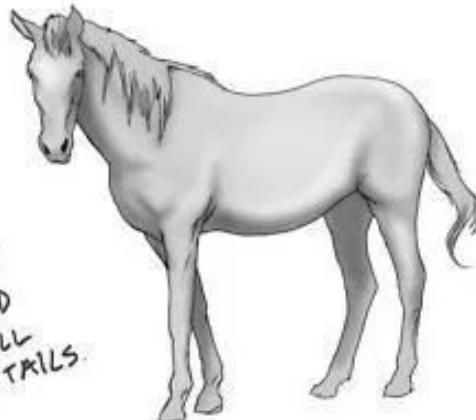


③ DRAW THE FACE



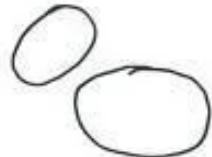
④ DRAW THE HAIR

⑤
ADD
SMALL
DETAILS



HOW TO:

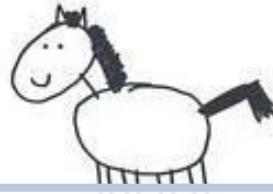
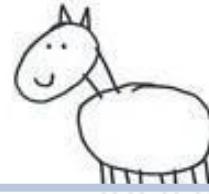
Take your prototype app to production



① DRAW 2 CIRCLE



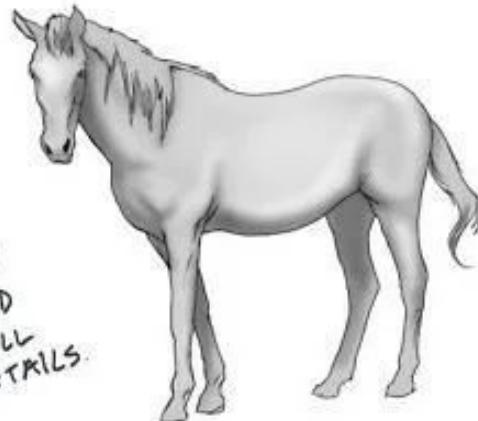
② DRAW THE LEGS



③ DRAW THE FACE

④ DRAW THE HAIR

⑤ ADD
SMALL
DETAILS



Prototype

Production

Demo!





- Large language model
- Vectorizer model
- Web app framework
- Vector search algorithm

Customer support analyst

Search

Query: Select the company account:

Limit: - + Search type: Hybrid Vector Keyword

Results

For query: `returns`

AmazonHelp: User_280559: We made a return and had previously b...

AmazonHelp: User_123163: if I buy something in black Friday am...



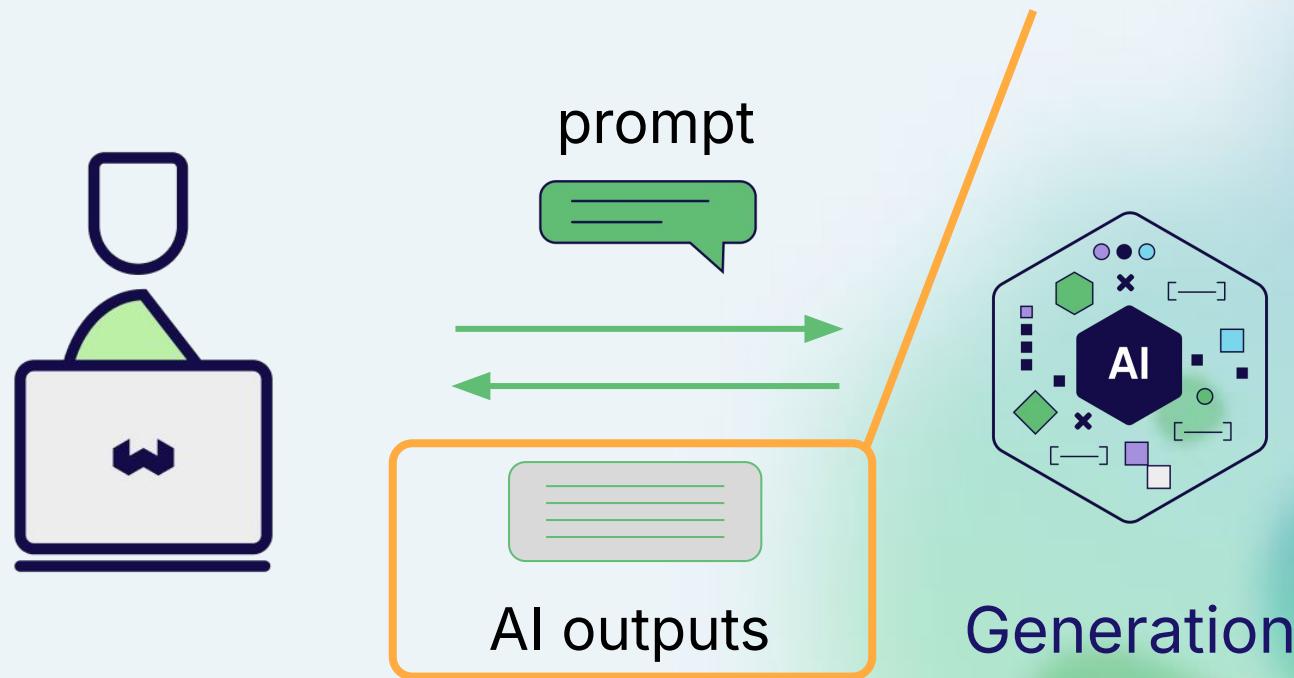
Recap

Databases & RAG (Retrieval Augmented Generation)



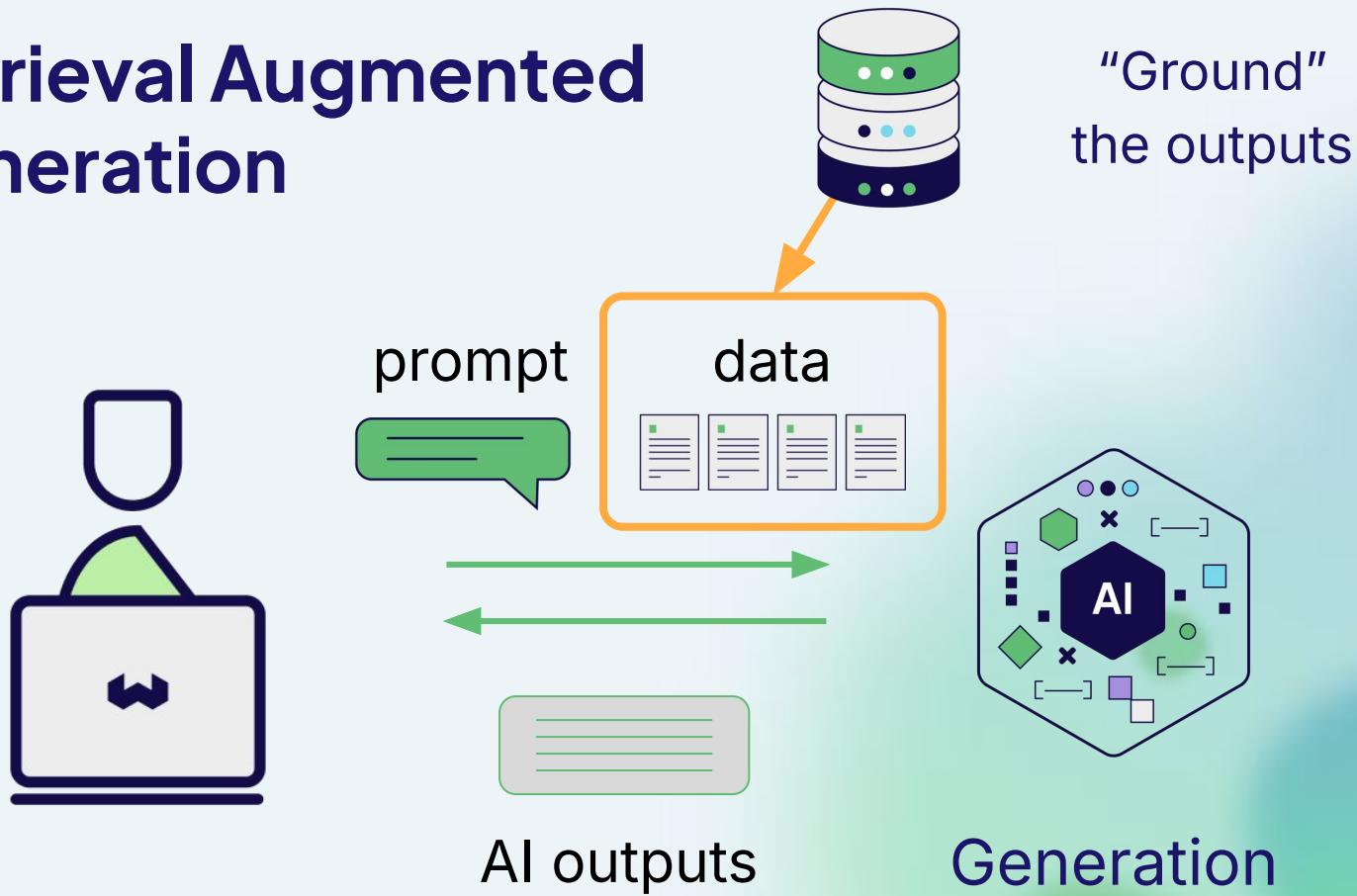
Large Language Model

Can “**hallucinate**”





Retrieval Augmented Generation





Gen AI: Prototyping to Production

Customer support analyst

Search

Query

returns

Select the company account

AmazonHelp

Limit

5

- +

Search type

Hybrid Vector Keyword

Results

For query: `returns`

AmazonHelp: User_280559: We made a return and had previously b...

AmazonHelp: User_123163: if I buy something in black Friday am...

Example source accounts: [AmazonHelp](#) (26351), [AppleSupport](#) (15397), [Uber_Support](#) (10876), [Delta](#) (6686), [AmericanAir](#) (6477), [comcastcares](#) (5569), [hulu_support](#) (4249), [SpotifyCares](#) (6055), [TMobileHelp](#) (4699), [SouthwestAir](#) (4931)

Cluster statistics

Object count

200000

Nodes

1



Gen AI Going to Production



Customer



Search

Query

returns

Limit

5

Results

For query: `ret`

AmazonHelp:

AmazonHelp: US

Select the company account

Great idea! But...

Hey, this prototype is popular!
Let's turn it into a spin-off!



Example source accounts: [AmazonHelp](#) (26351), [AppleSupport](#) (15397), [Coca_Cola_Support](#) (10876), [Delta](#) (6686), [American_Airlines](#) (5577), [comcastcares](#) (5569), [hulu_support](#) (4577), [Kia_Cares](#) (6055), [TMobileHelp](#) (4031), [Verizon](#) (381)



Cluster statistics

Object count

200000

Nodes

1



Gen



Going to Productio

Hey, this prototype is popular!

Let's turn it into a spin-off!

Search

Query

returns

Limit

5

Results

For query: `ret`

AmazonHelp:

AmazonHelp: US

Select the company account

How do we:

- Deal with 1000x more data?
- Meet regulatory requirements?
- Reduce downtime?
- AND make money?



Example source accounts: AmazonHelp (26351), AppleSupport (15397), Cerner_Support (10876), Delta (6686), AmericanAirlines (4777), comcastcares (5569), hulu_support (4000), ComcastCares (6055), TMobileHelp (4000), USAA (381)



Cluster statistics

Object count

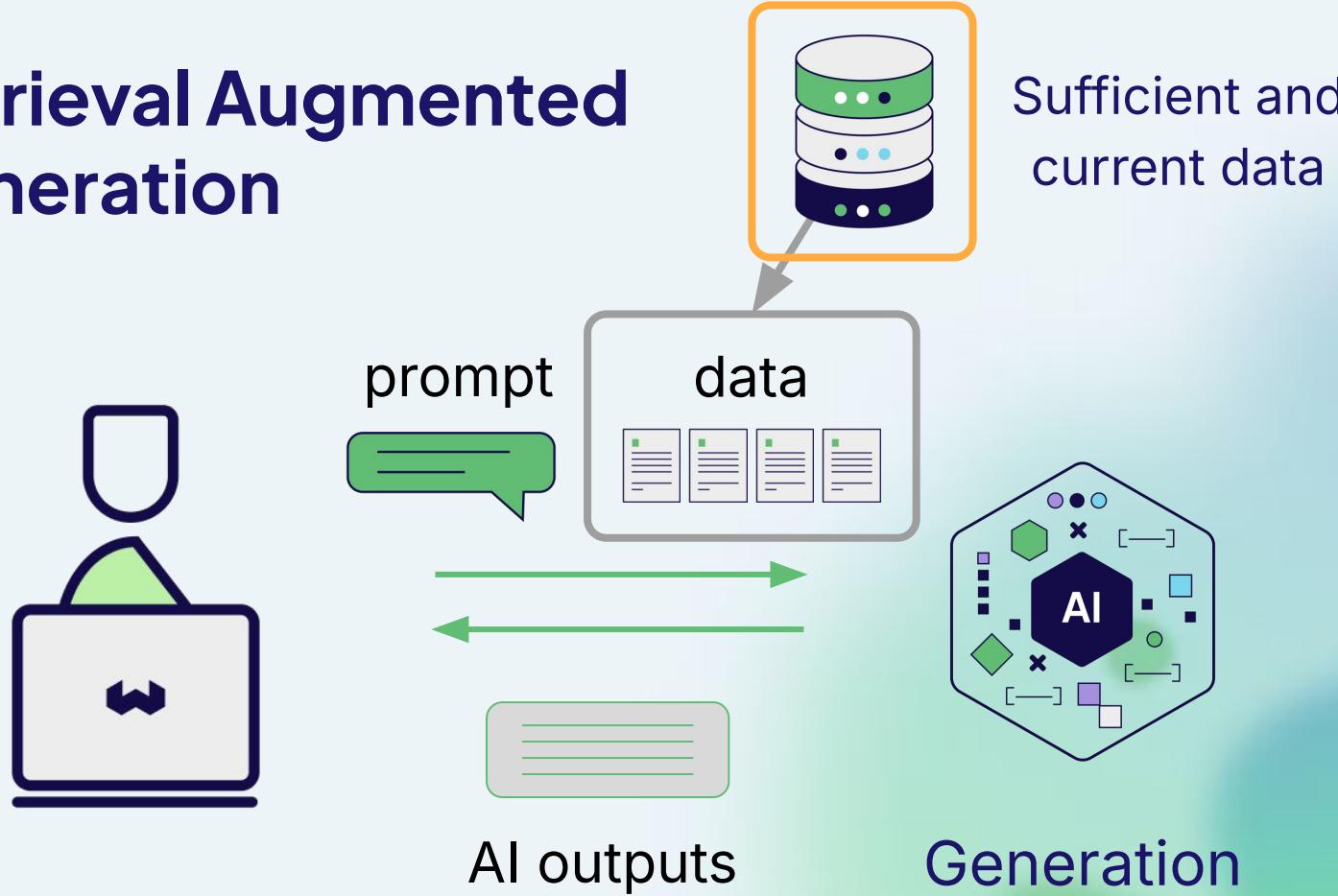
200000

Nodes

1



Retrieval Augmented Generation





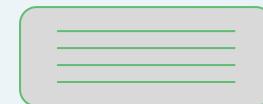
Retrieval Augmented Generation



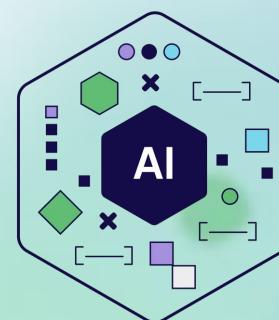
prompt



data



AI outputs



Generation



Challenges

Dealing with scale
Search speed



Gen AI: Prototyping to Production

Customer support analyst

Search

Query

returns

Select the company account

AmazonHelp

Limit

5

- +

Search type

Hybrid Vector Keyword

Results

For query: `returns`

AmazonHelp: User_280559: We made a return and had previously b...

AmazonHelp: User_123163: if I buy something in black Friday am...

Example source accounts: [AmazonHelp](#) (26351), [AppleSupport](#) (15397), [Uber_Support](#) (10876), [Delta](#) (6686), [AmericanAir](#) (6477), [comcastcares](#) (5569), [hulu_support](#) (4249), [SpotifyCares](#) (6055), [TMobileHelp](#) (4699), [SouthwestAir](#) (4931)

Cluster statistics

Object count

200000

Nodes

1



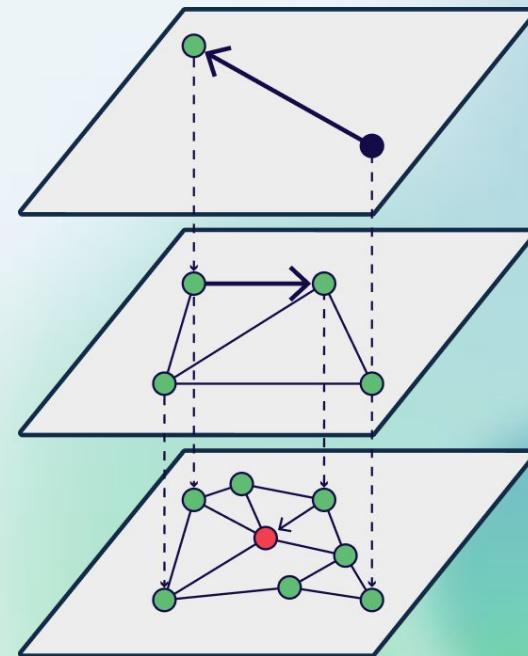
Scale: Solutions

Improve efficiency - indexing

Scale: Solutions

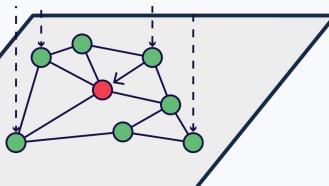
Improve efficiency - indexing

- Default: **HNSW** index

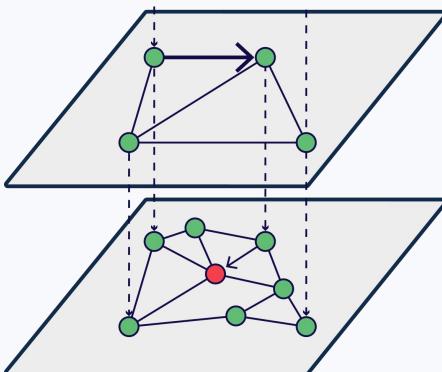


HNSW in focus

- Layered graphs
- Lowest layer: all vectors (nodes)
 - Nodes connected to neighbors



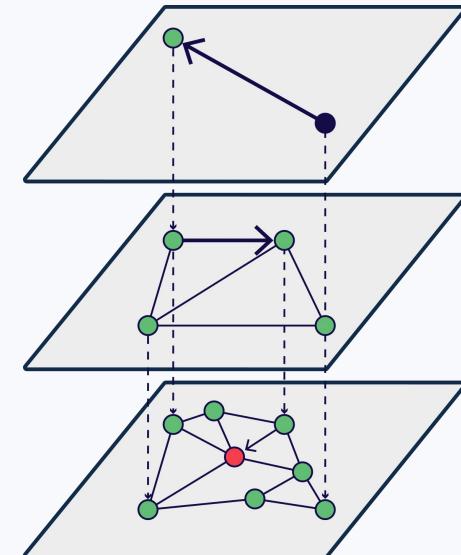
HNSW



HNSW

HNSW in focus

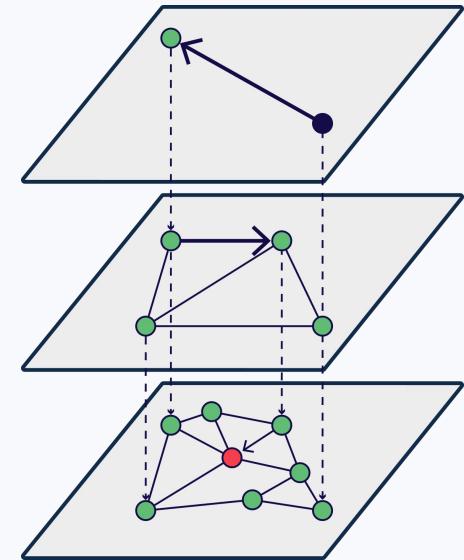
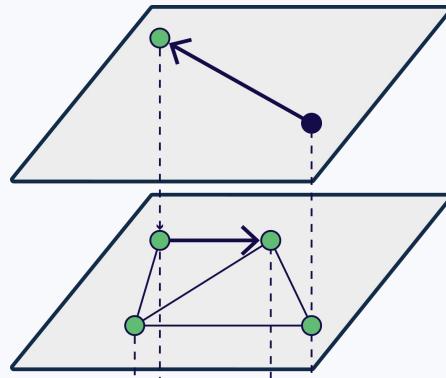
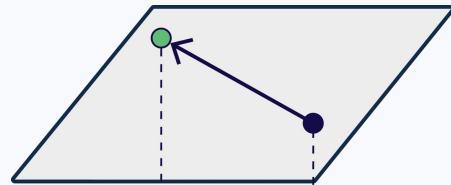
- Layered graphs
- Lowest layer: all vectors (nodes)
 - Nodes connected to neighbors
- Upper layers: subsets of vectors



HNSW

HNSW in focus

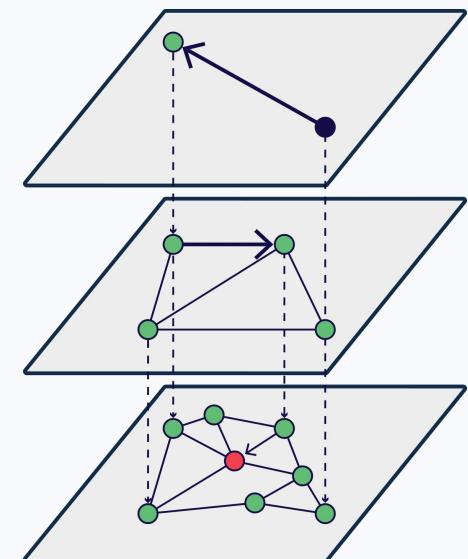
- Layered graphs
- Lowest layer: all vectors (nodes)
 - Nodes connected to neighbors
- Upper layers: subsets of nodes
 - Higher \leftrightarrow fewer nodes
 - Higher \leftrightarrow faster traversal
- Multi-dimensional skip lists



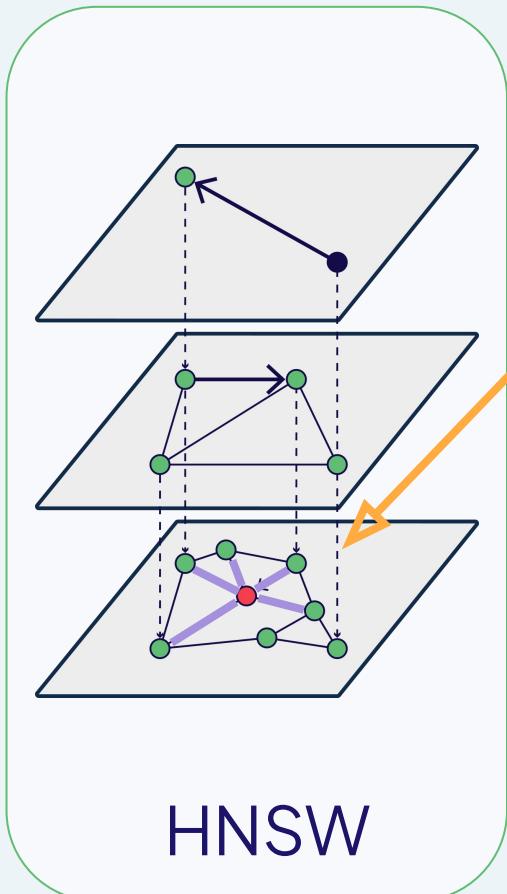
HNSW



HNSW parameters



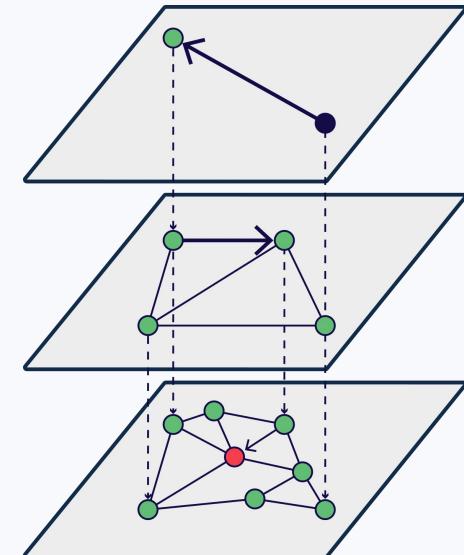
HNSW



HNSW parameters

- **maxConnections (m)**

- N (node → node) connections
 - e.g. 32
- Note: bottom layer → $m * 2$

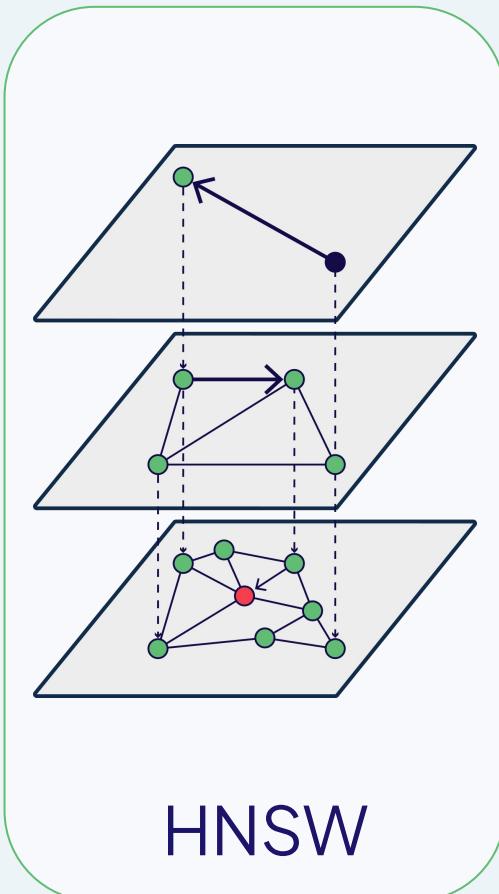


HNSW

HNSW parameters

- maxConnections (m)
- **ef** → size of list during search
 - Higher → better
 - Cost: slower

rank	object	distance
1	<uuid>	0.0514
2	<uuid>	0.0642
...
...
...
...
max(ef, limit)	<uuid>	0.3532



HNSW parameters

- maxConnections (m)
- ef
- **efConnections**
 - For graph construction
 - Higher → better
Slower insertion



Solutions

Search speed

- Vector index
 - HNSW
 - Logarithmic scaling
 - Scalable, fast search
 - Configurable parameters



Challenges

Dealing with scale
Resource management



Search

Query Select the company account

Limit - + Search type Hybrid Vector Keyword

Results

For query: `returns`

AmazonHelp: User_280559: We made a return and had previously b...

AmazonHelp: User_123163: if I buy something in black Friday am...

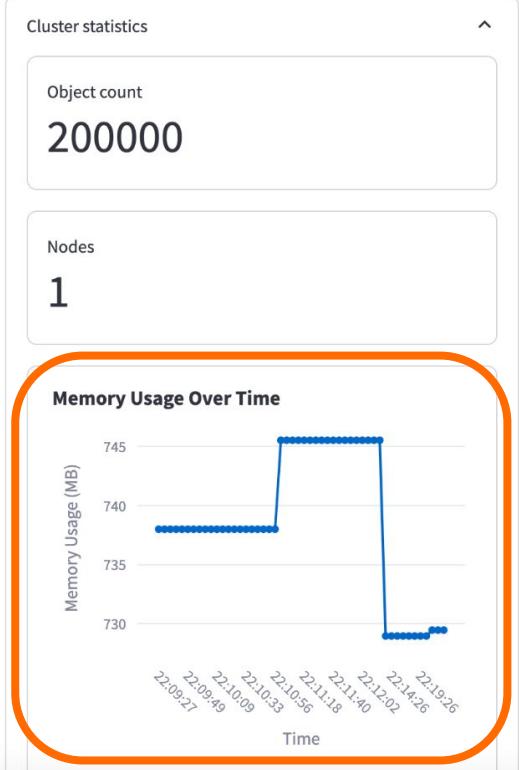
AmazonHelp: User_247957: I didn't know a product didn't have f...

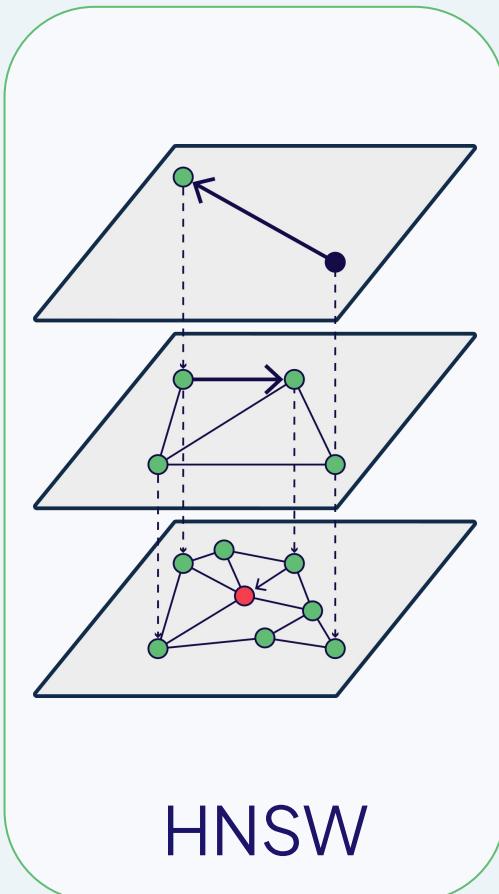
RAG

What should we do with the search results?

What patterns do you recognize in these comments?

i Example source accounts: AmazonHelp (26351), AppleSupport (15397), Uber_Support (10876), Delta (6686), AmericanAir (6477), comcastcares (5569), hulu_support (4249), SpotifyCares (6055), TMobileHelp (4699), SouthwestAir (4931)





HNSW in focus

- In-memory data structure
- Typical limitation: memory footprint
 - Connections (~10B/ea)
 - Vectors (~4B/dim; 6kB/vector)
 - @10M objects
 - ~65GB!
 - @100M → 650GB, etc...



🔍 1024 gib

X

19 matches

Instance
name

On-Demand
hourly rate

x2gd.metal

\$5.344

x2gd.16xlarge

hpc6id.32xlarge

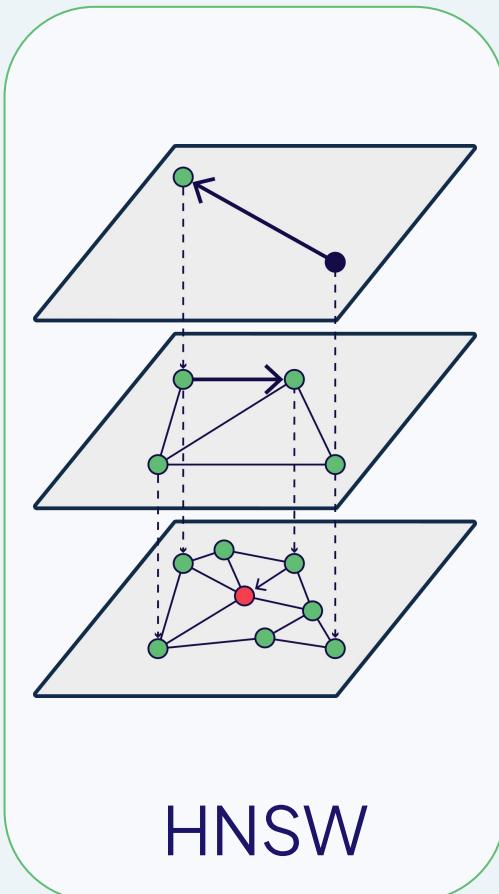
x2iedn.8xlarge

x2idn.16xlarge



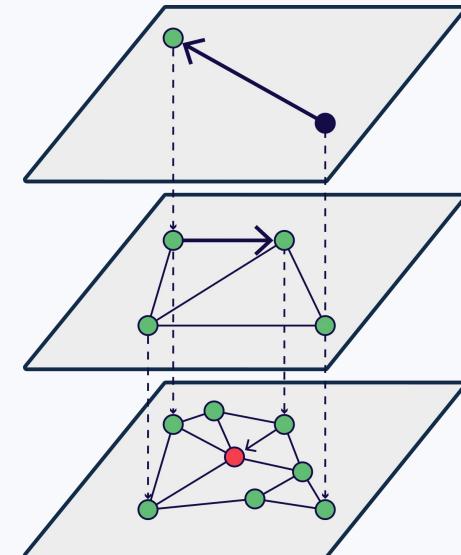
$$120 * 365 = \$43,800 / \text{year!}$$

AWS Spot pricing, Nov 2024



HNSW in focus

- In-memory data structure
- Typical limitation: memory footprint
 - Connections ($\sim 10B/\text{ea}$)
 - **Vectors ($\sim 4B/\text{dim}$; $6kB/\text{vector}$)**
 - @10M objects
 - ~65GB!
 - @100M \rightarrow 650GB, etc...



HNSW

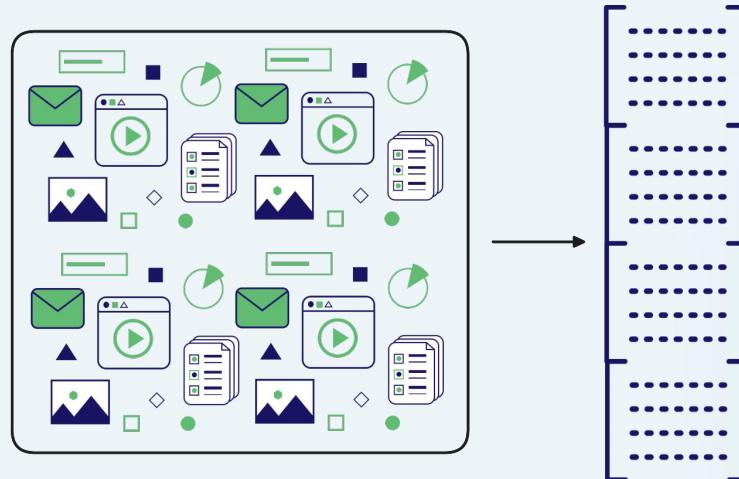
Quantization

- Compression
- Reduce precision from floats
 - Binary / scalar / locally adaptive
- Reduce number of dimensions
 - Product quantization



Scale: Solutions

Improve efficiency

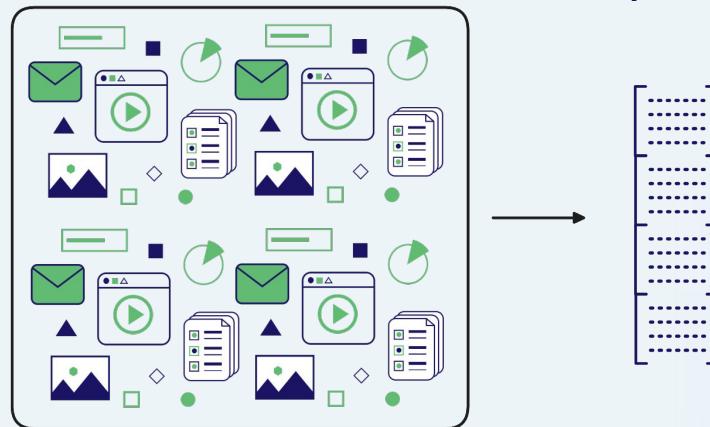




Scale: Solutions

Improve efficiency

Compression

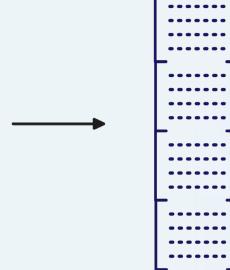


Scale: Solutions

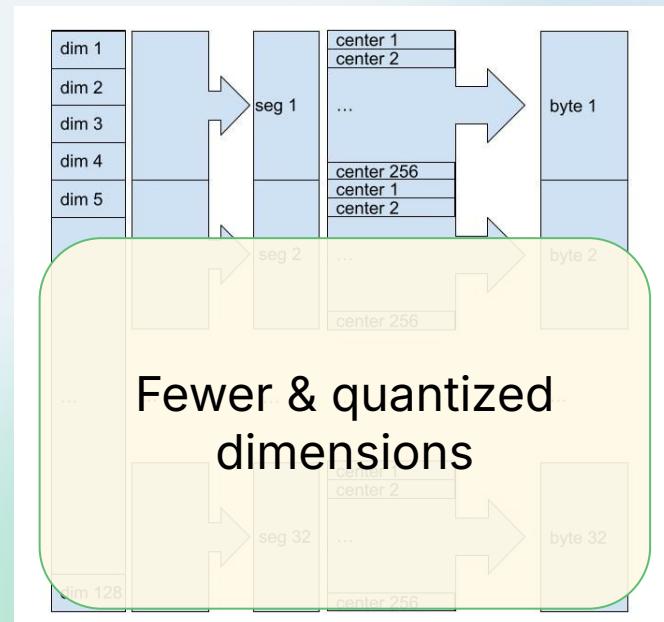
Improve efficiency



Compression

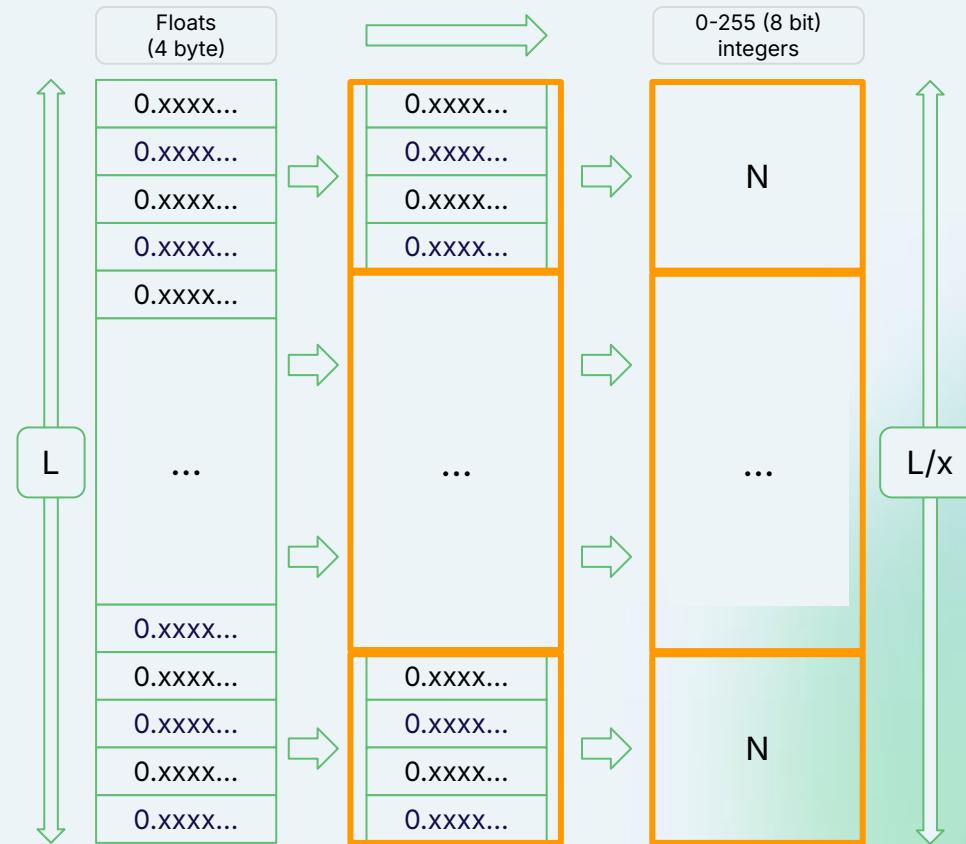


Product quantization



Customisable compression

(e.g. 128 floats → 32 bytes: 16x)





Scale: Solutions

Improve efficiency



Compression



Binary quantization

$[-0.1324..., -0.9253...,$
 $0.2389...,$
 \dots
 $0.3249..., 0.2390..., -0.4823...]$



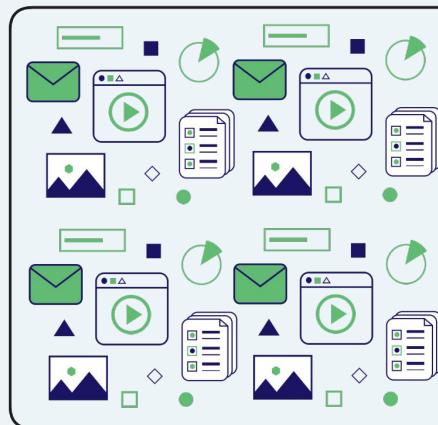
$[0, 0, 1, 0, 1, 1,$
 \dots
 $0, 1, 0, 1, 1, 0]$

$n \text{ floats} \rightarrow n \text{ bits}$
(32x reduction)



Scale: Solutions

Improve efficiency



Compression



Scalar quantization

$[-0.1324..., -0.9253...,$
 $0.2389...,$
 \dots
 $0.3249..., 0.2390..., -0.4823...]$



$[104, 12, 138,$
 \dots
 $152, 138, 47]$

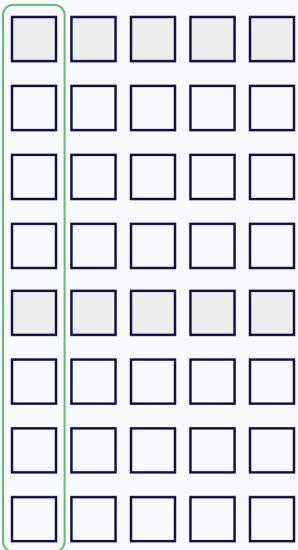
$n \text{ floats} \rightarrow n \text{ ints}$
(4x reduction)



Scale: Solutions

Binary quantization

```
articles = client.collections.create(  
    name="Article",  
    properties=[  
        Property(name="title", data_type=DataType.TEXT)  
    ],  
    vector_index_config=wc.Configure.VectorIndex.flat(  
        quantizer=wc.Configure.VectorIndex.Quantizer.bq()  
    ),  
)
```



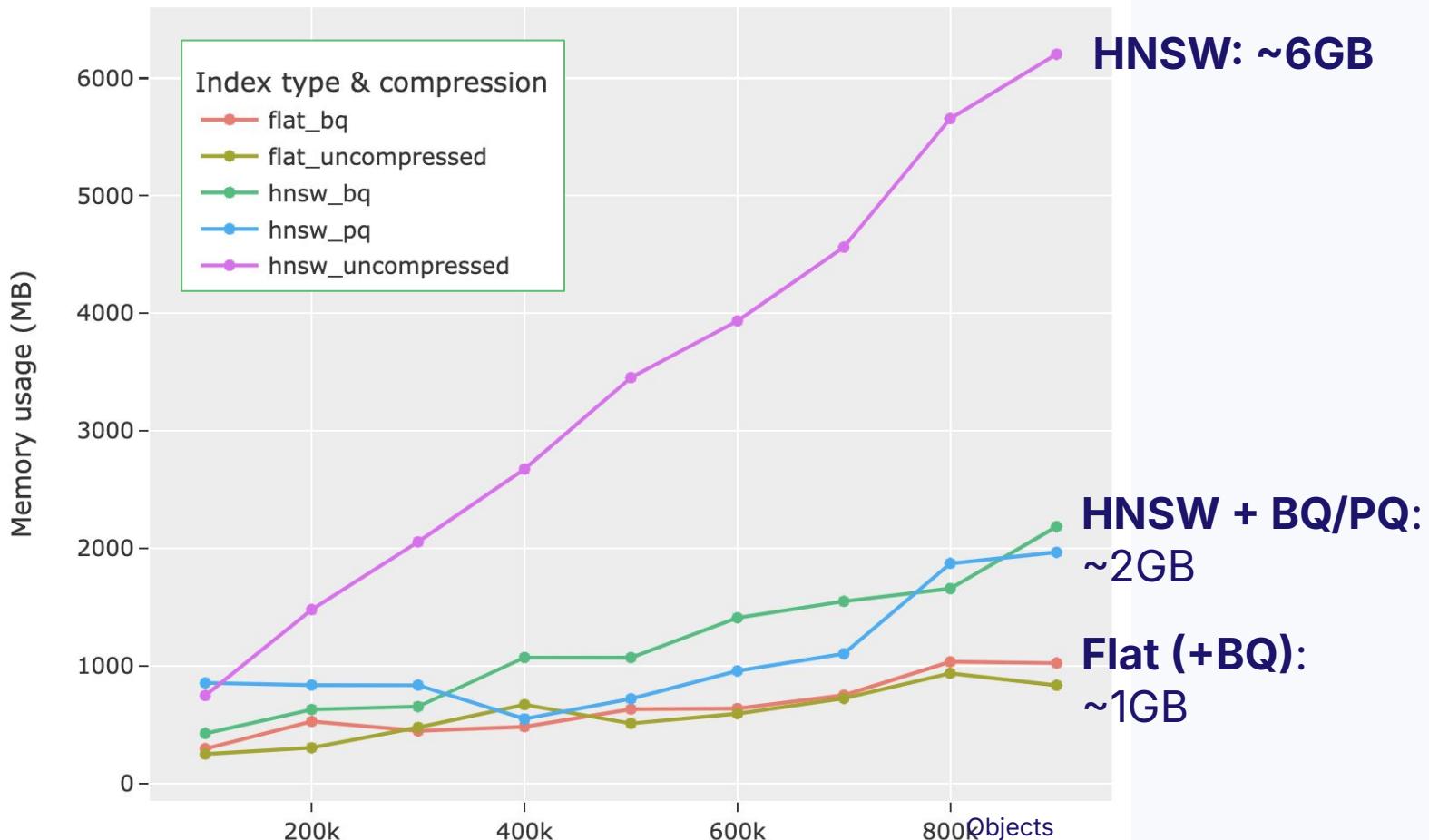
Flat

Flat index

- Simple data structure (e.g. map)
- Low overhead
- Does not scale well
 - Linear search
 - Good for small sets



Example: Index type & quantization vs memory footprint





Instance type

Memory

\$58k / year

View



VS.

\$23k / year

vCPU

32

< 1 2 >

Instance name	Demand hourly rate	vCPU	Memory	Storage	Network performance
x2iedn.8xlarge	\$6.669	32	1024 GiB	1 x 950 NVMe SSD	25 Gigabit
x1e.8xlarge	\$6.672	32	976 GiB	1 x 960 SSD	Up to 10 Gigabit
x2gd.8xlarge	\$2.672	32	512 GiB	1 x 1900 SSD	12 Gigabit

AWS Spot pricing, Nov 2024



Scale: Solutions

BQ / PQ / SQ compression

Search **quality** mitigated by over-fetching & rescoreing

When to use which?

- Generally, try PQ first
- BQ: model-specific





Solutions

Memory usage

- Quantization
 - Product, binary, scalar...
 - Trade off memory vs quality
 - Can also speed up search
- Flat index for small subsets



Challenges

Dealing with scale
Scaling (horizontal)



Search

Query Select the company account

Limit - + Search type Hybrid Vector Keyword

Results

For query: `returns`

AmazonHelp: User_280559: We made a return and had previously b...

AmazonHelp: User_123163: if I buy something in black Friday am...

AmazonHelp: User_247957: I didn't know a product didn't have f...

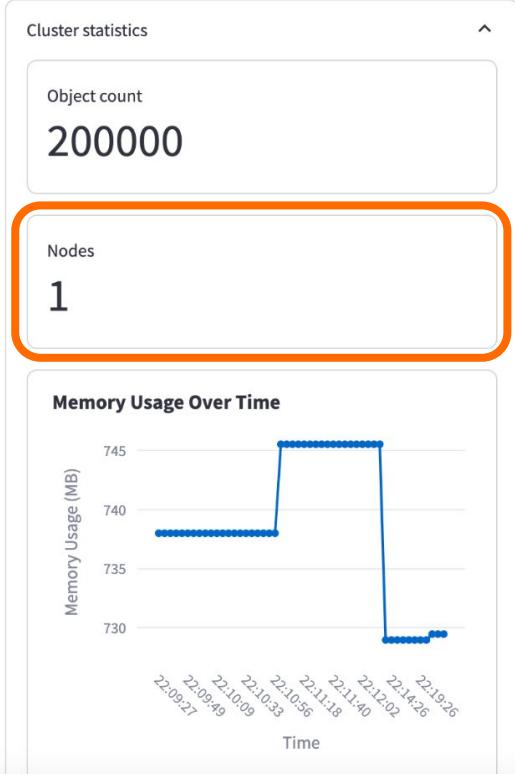
RAG

What should we do with the search results?

What patterns do you recognize in these comments?

Generate response

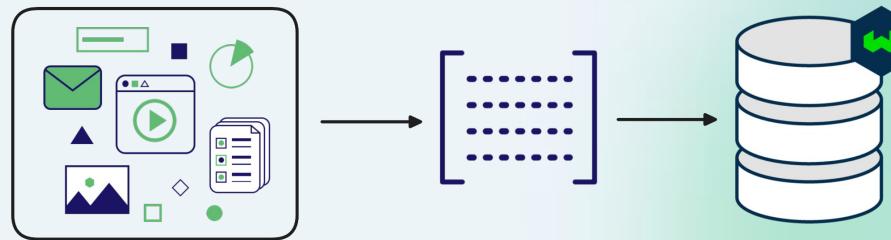
i Example source accounts: AmazonHelp (26351), AppleSupport (15397), Uber_Support (10876), Delta (6686), AmericanAir (6477), comcastcares (5569), hulu_support (4249), SpotifyCares (6055), TMobileHelp (4699), SouthwestAir (4931)





Scale: Solutions

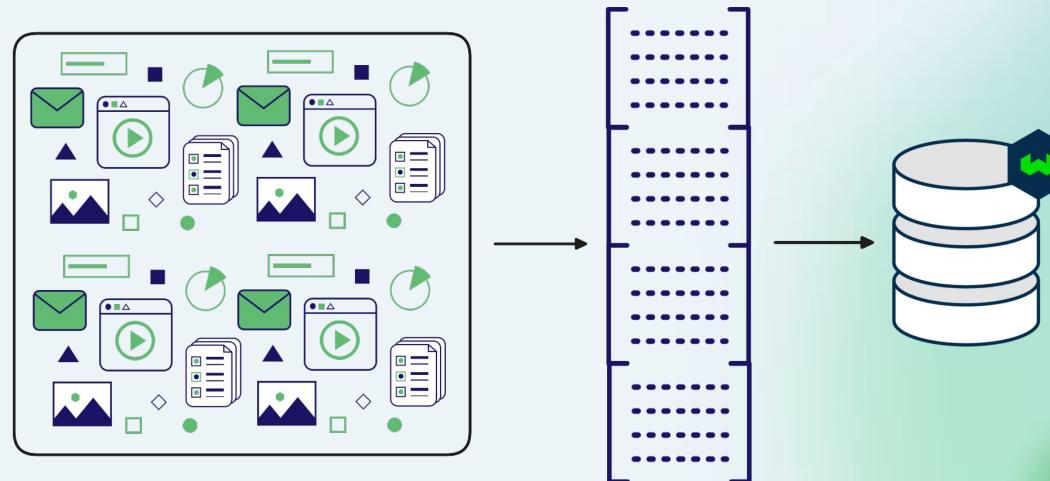
Growth





Scale: Solutions

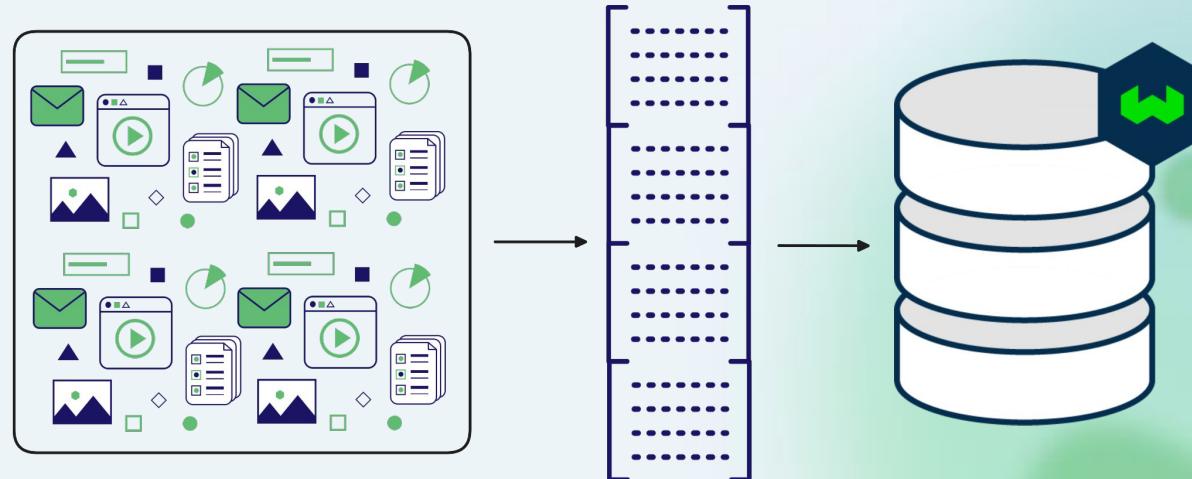
Growth





Scale: Solutions

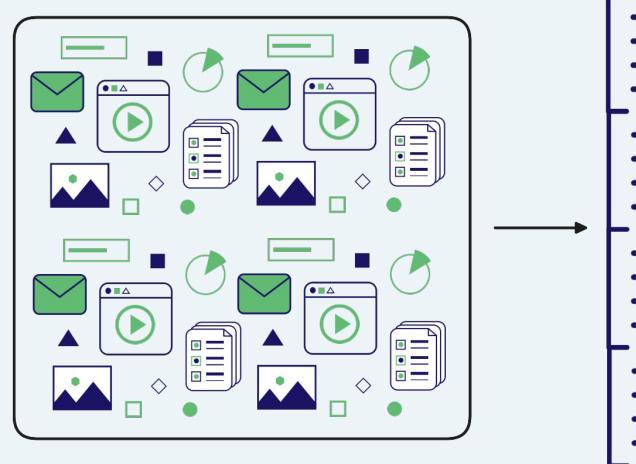
Growth





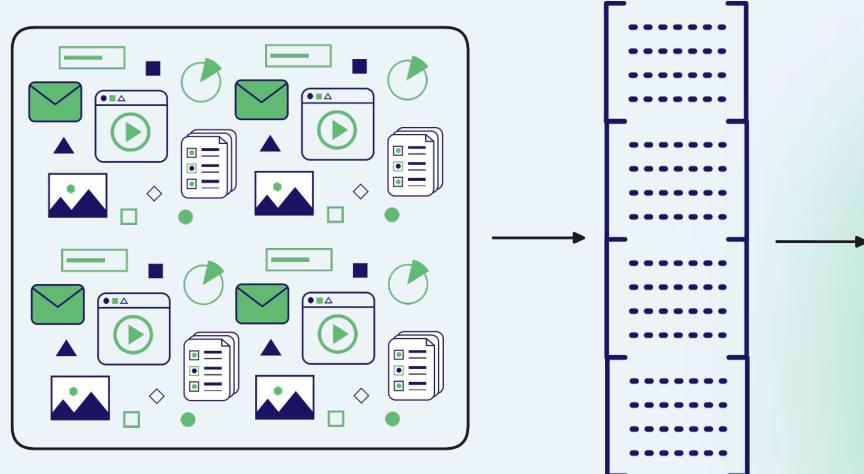
Scale: Solutions

Growth



Scale: Solutions

Growth



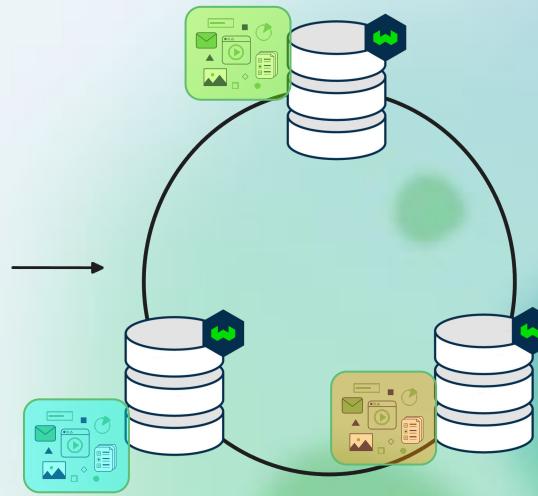


Scale: Solutions

Deal with growth



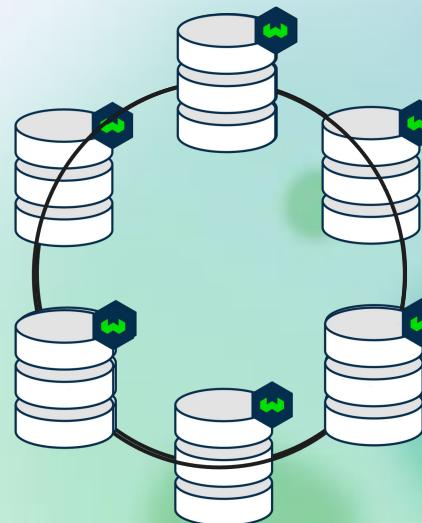
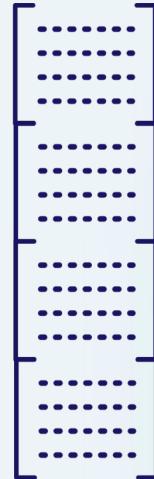
Horizontal scaling
(sharding)





Scale: Solutions

Deal with growth



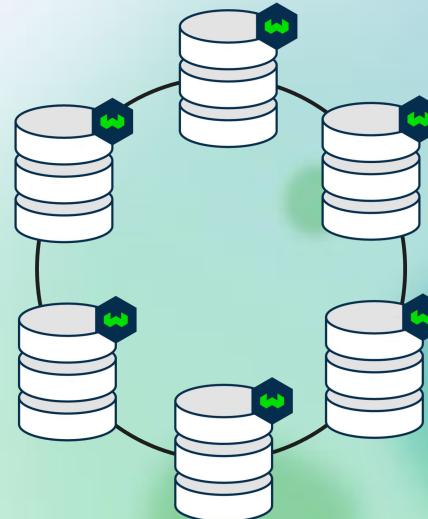
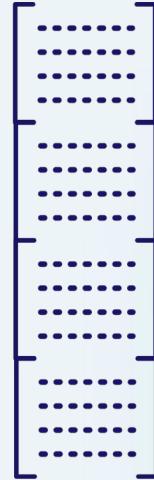
To scale:
Add more nodes

Billion scale
since 2022!



Scale: Solutions

Deal with growth



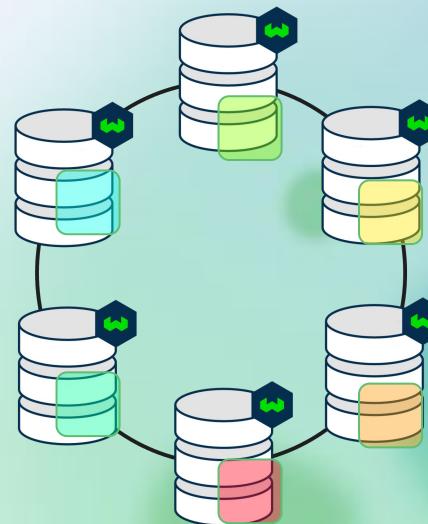
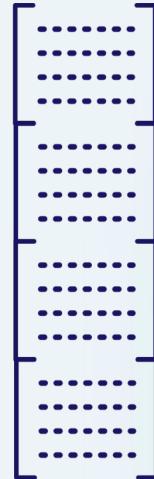
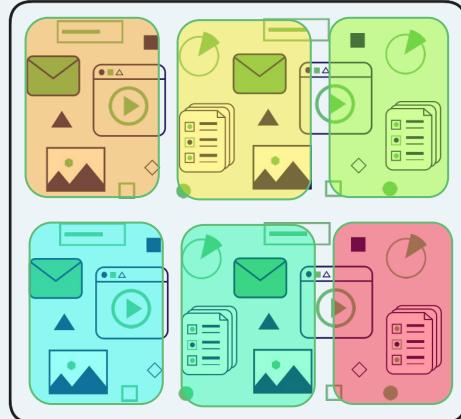
To scale:
Add more nodes

Billion scale
since 2022!



Scale: Solutions

Deal with growth



To scale:
Add more nodes

Billion scale
since 2022!



Solutions

Scalability

- Horizontal scaling (sharding)
 - Shard vector + inverted indexes
 - Scale out as needed



Challenges

Dealing with multiple users
Isolation



Search

Query

returns

Limit

5

- +

Select the company account

AmazonHelp

Search type

Hybrid Vector Keyword



Example source accounts: AmazonHelp (26351), AppleSupport (15397), Uber_Support (10876), Delta (6686), AmericanAir (6477), comcastcares (5569), hulu_support (4249), SpotifyCares (6055), TMobileHelp (4699), SouthwestAir (4931)

Results

For query: `returns`

AmazonHelp: User_280559: We made a return and had previously b...

AmazonHelp: User_123163: if I buy something in black Friday am...

AmazonHelp: User_247957: I didn't know a product didn't have f...

RAG

What should we do with the search results?

What patterns do you recognize in these comments?

Generate response

Cluster statistics

Object count

200000

Nodes

1

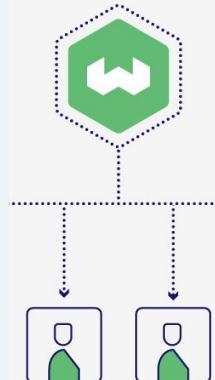
Memory Usage Over Time





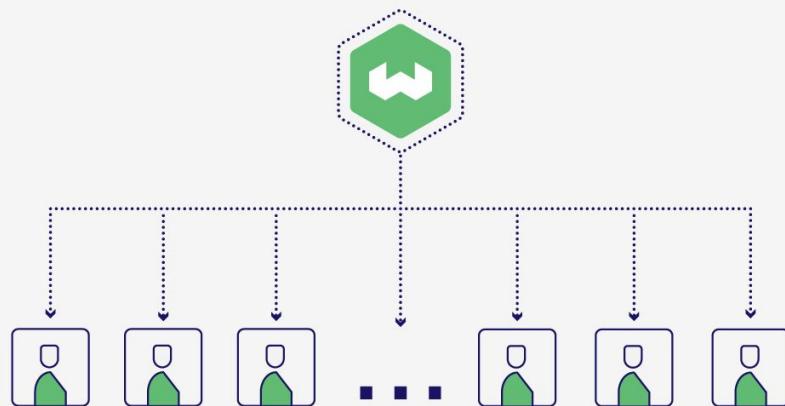
Scale: Solutions

End user growth



Scale: Solutions

End user growth



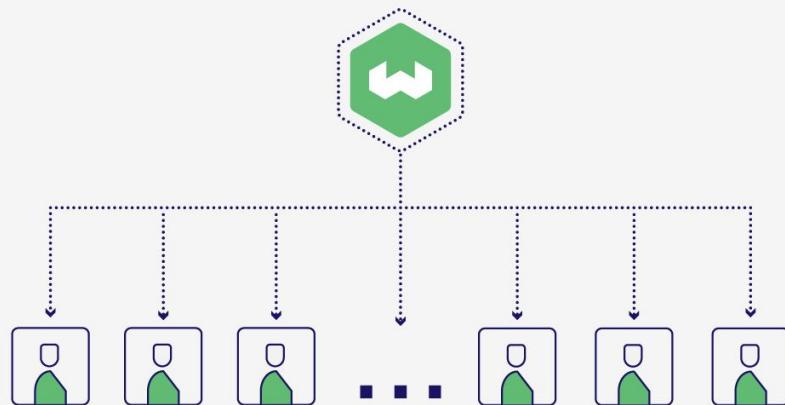
Challenges faced:

- Performance
- Data isolation
- Compliance

Developed: **Multi-tenancy**

Scale: Solutions

End user growth

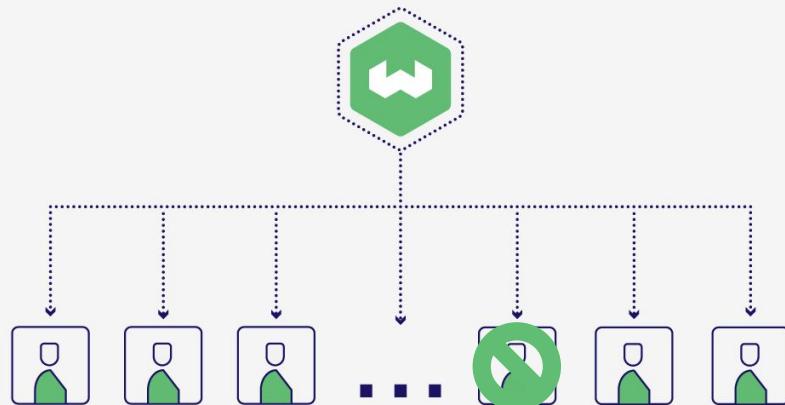


Multi-tenancy implementation

- 1000s per node
- Isolated
- Tenant state management

Scale: Solutions

End user growth

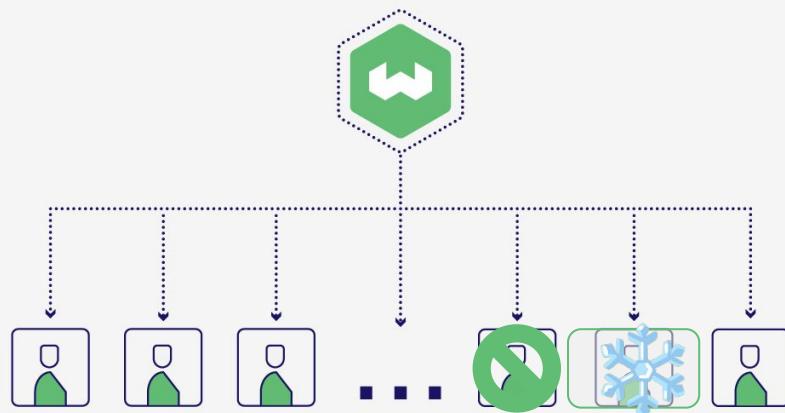


Multi-tenancy implementation

- 1000s per node
- **Isolated** (easy deletion & compliance)
- Tenant state management

Scale: Solutions

End user growth



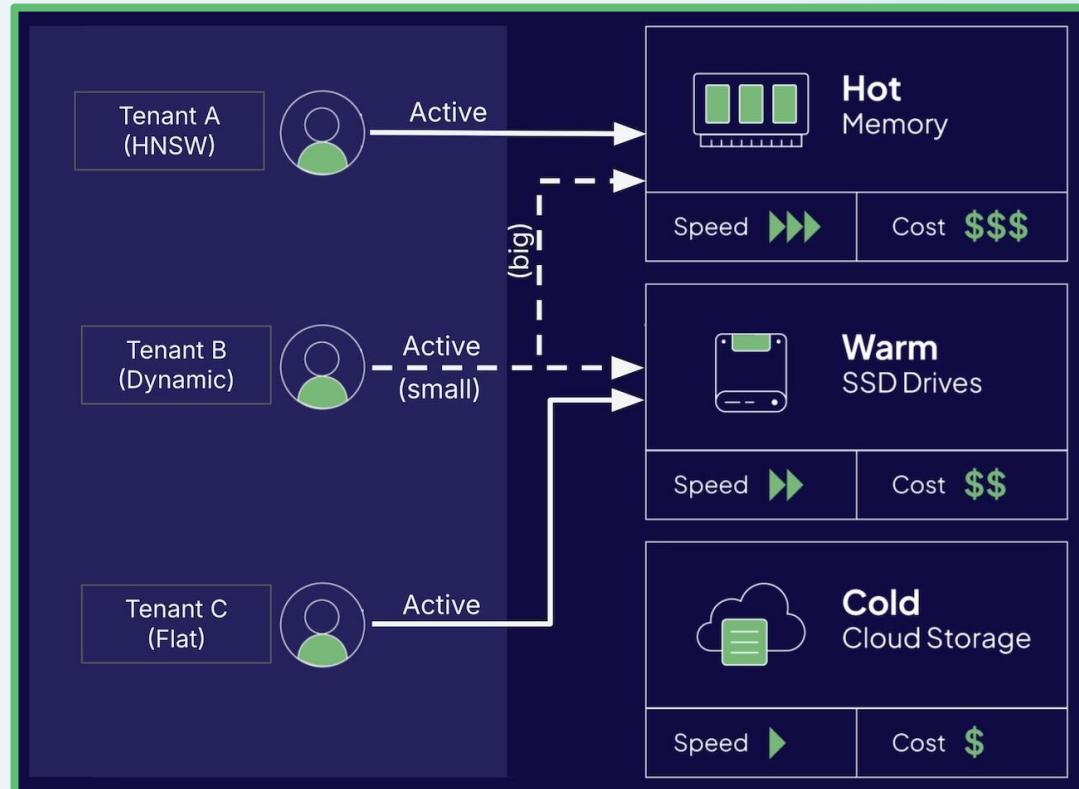
Multi-tenancy implementation

- 1000s per node
- Isolated
- **Tenant state management**
 - Active/inactive/offloaded



Scale: Solutions

Storage tiers





Solutions

Isolation

- Multi-tenancy
 - Isolated shards
 - Shard per end user
 - Easy on/offboarding
 - Manage resource usage



Challenges

Dealing with reliability
Fault tolerance



Search

Query

returns

Select the company account

AmazonHelp

Limit

5

- +

Search type

Hybrid Vector Keyword

Results

For query: `returns`

AmazonHelp: User_280559: We made a return and had previously b...

AmazonHelp: User_123163: if I buy something in black Friday am...

AmazonHelp: User_247957: I didn't know a product didn't have f...

RAG

What should we do with the search results?

What patterns do you recognize in these comments?

Generate response

i Example source accounts: AmazonHelp (26351), AppleSupport (15397), Uber_Support (10876), Delta (6686), AmericanAir (6477), comcastcares (5569), hulu_support (4249), SpotifyCares (6055), TMobileHelp (4699), SouthwestAir (4931)

Cluster statistics

Object count

200000

Nodes

1

Memory Usage Over Time





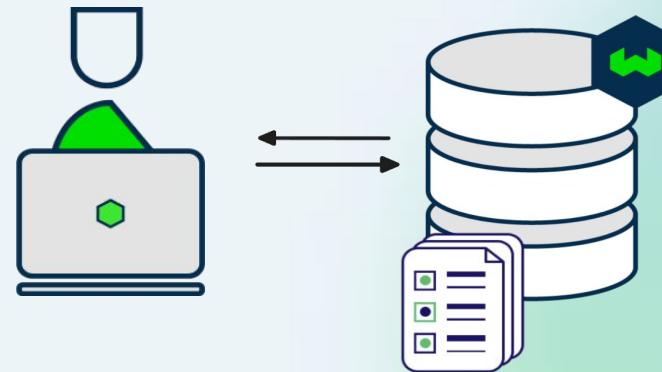
Reliability: Considerations

- **Robust to errors:** Ensure consistency
- **Downtime:** Reduce disruption
- **Backups:** In case of emergency



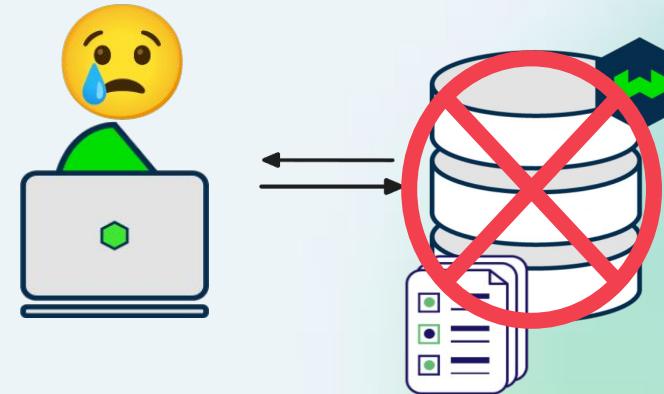
Reliability: Solutions

Happy days



Reliability: Solutions

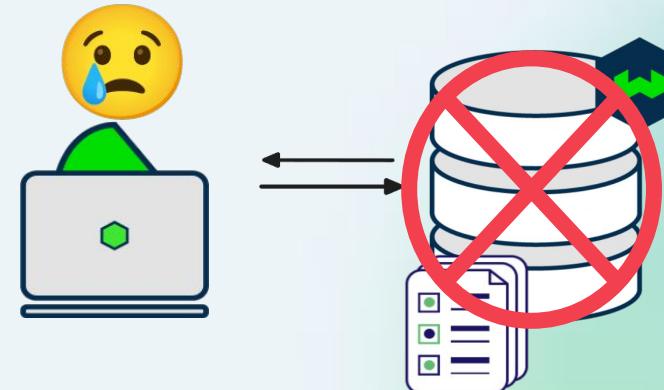
(Less) Happy days





Reliability: Solutions

(Less) Happy days



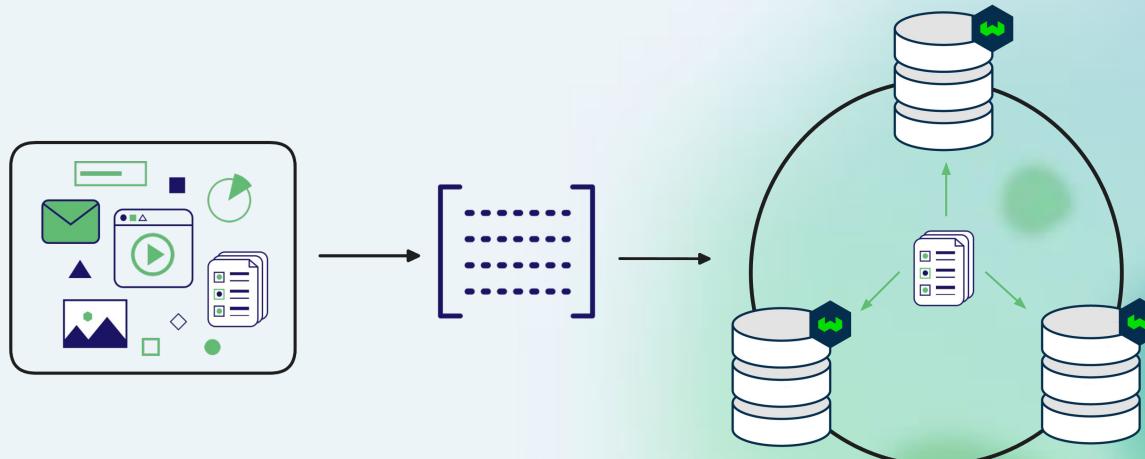
Downtime = inevitable



Reliability: Solutions

Provide redundancy

Replication

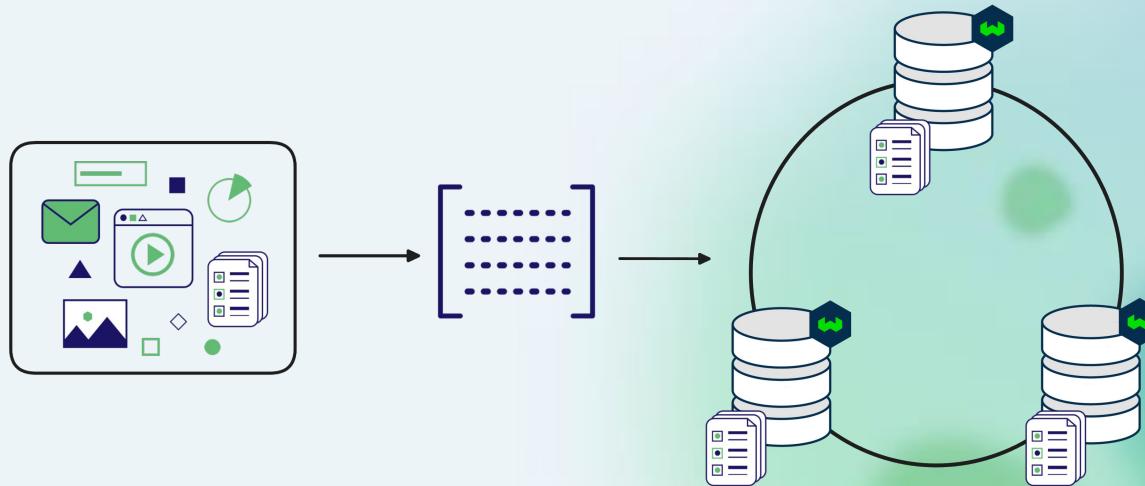




Reliability: Solutions

Provide redundancy

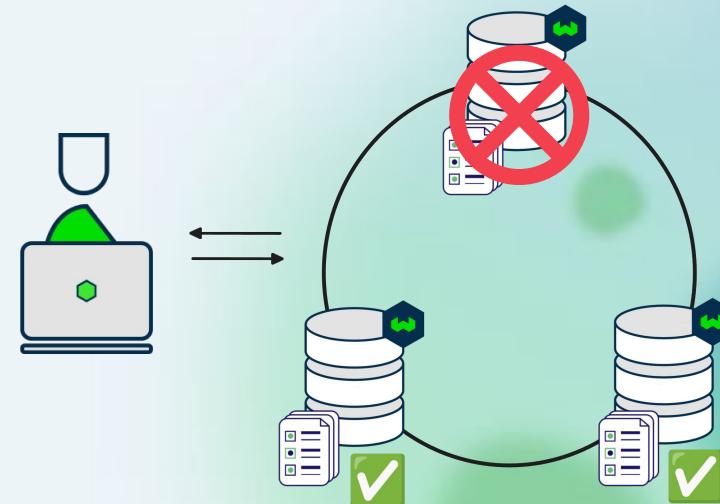
Replication



Reliability: Solutions

Provide redundancy

Replication



Reliability: Solutions

Provide redundancy

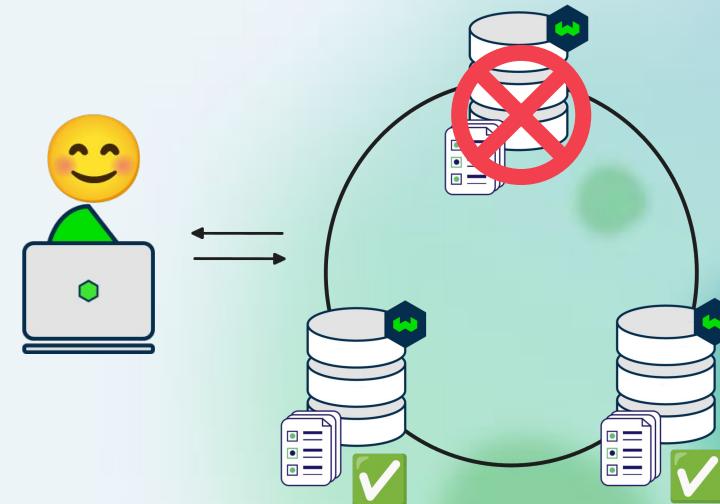
Node-level downtime:

- inevitable

System-level downtime:

- avoidable!

Replication





Reliability: Solutions

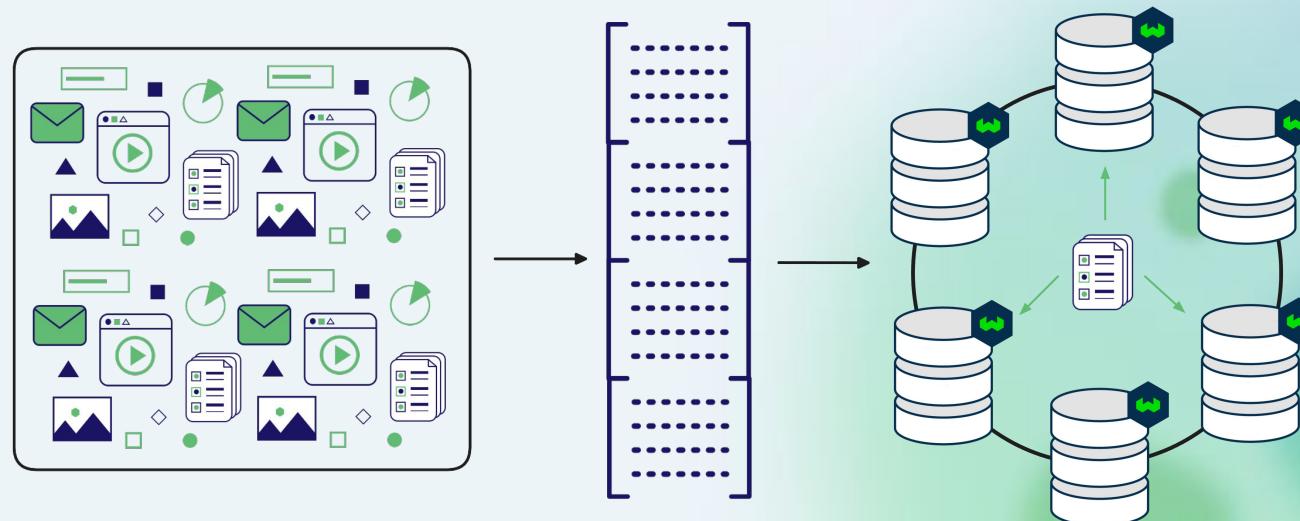


```
articles = client.collections.create(  
    name="Article",  
    properties=[  
        Property(name="title", data_type=DataType.TEXT)  
    ],  
    replication_config=Configure.replication(factor=3)  
)
```



Reliability: Solutions

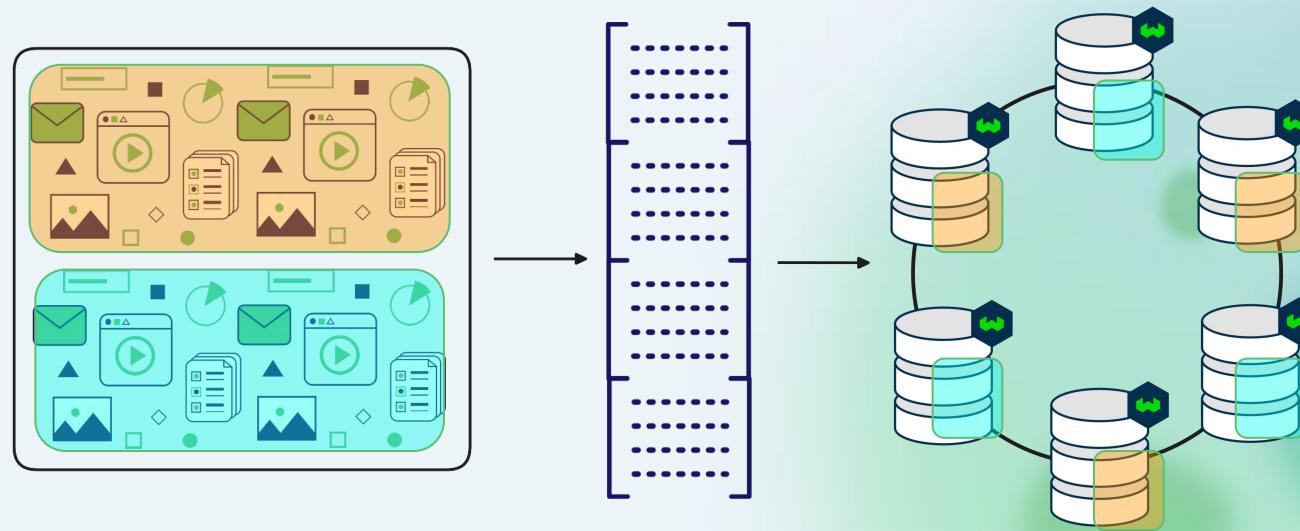
Sharding + replication





Reliability: Solutions

Sharding + replication





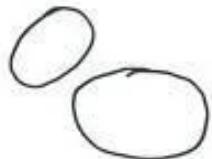
Solutions

Fault tolerance

- Replication
 - Data replication
 - Tunable consistency
 - Combine with multi-tenancy / horizontal scaling

HOW TO:

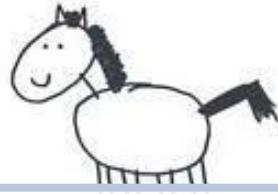
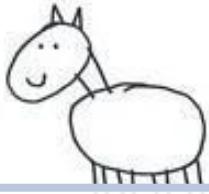
Take your prototype app to production



① DRAW 2 CIRCLES



② DRAW THE LEGS



Prototype

③ DRAW THE FACE

④ DRAW THE HAIR

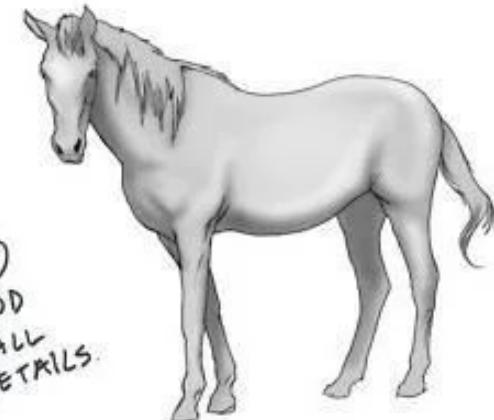
Vector index types

Vector quantization

Sharding

Multi-tenancy

Replication



⑤ ADD
SMALL
DETAILS.

Production



Thank you

Slides &
demo app



weaviate.io



[weaviate/weaviate](https://github.com/weaviate/weaviate)



jphwang