zilliz

Exploring Vector Search at Scale

Stephen Batifol

Haystack EU 24

The Happy Beginnings

zilliz

The Tech Stack in Q4 2024

CURSOR

Claude

zilliz

Unstructured Data? No Problem!

- Docker + Vector DB
- Use some Embedding Model
- Store your Vectors in your DB, you're happy! :D

zilliz

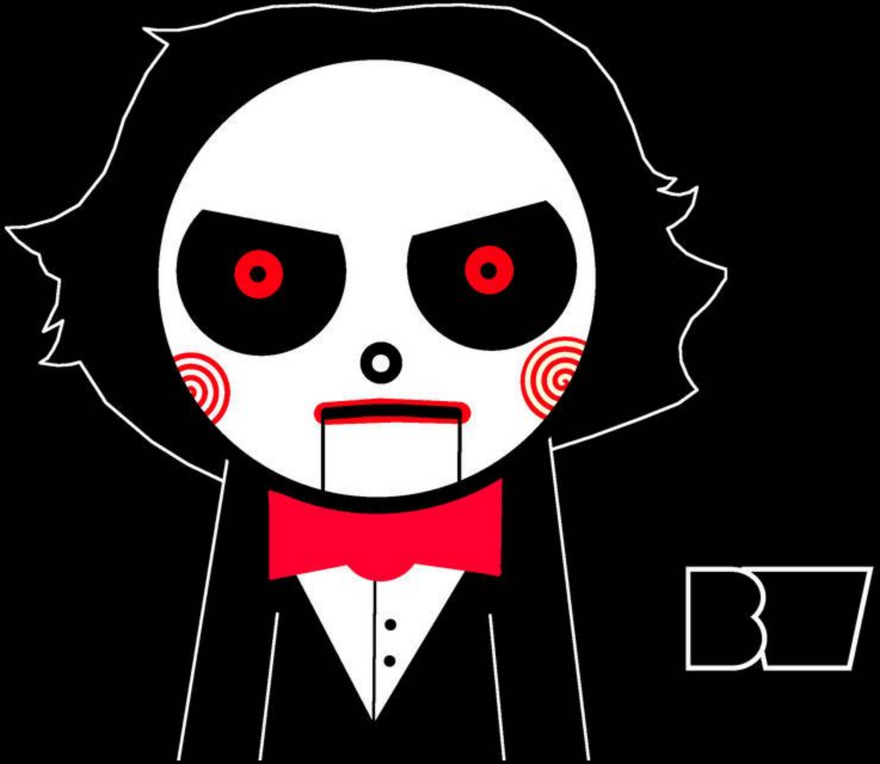
The Growing Pains

You have more and more data!

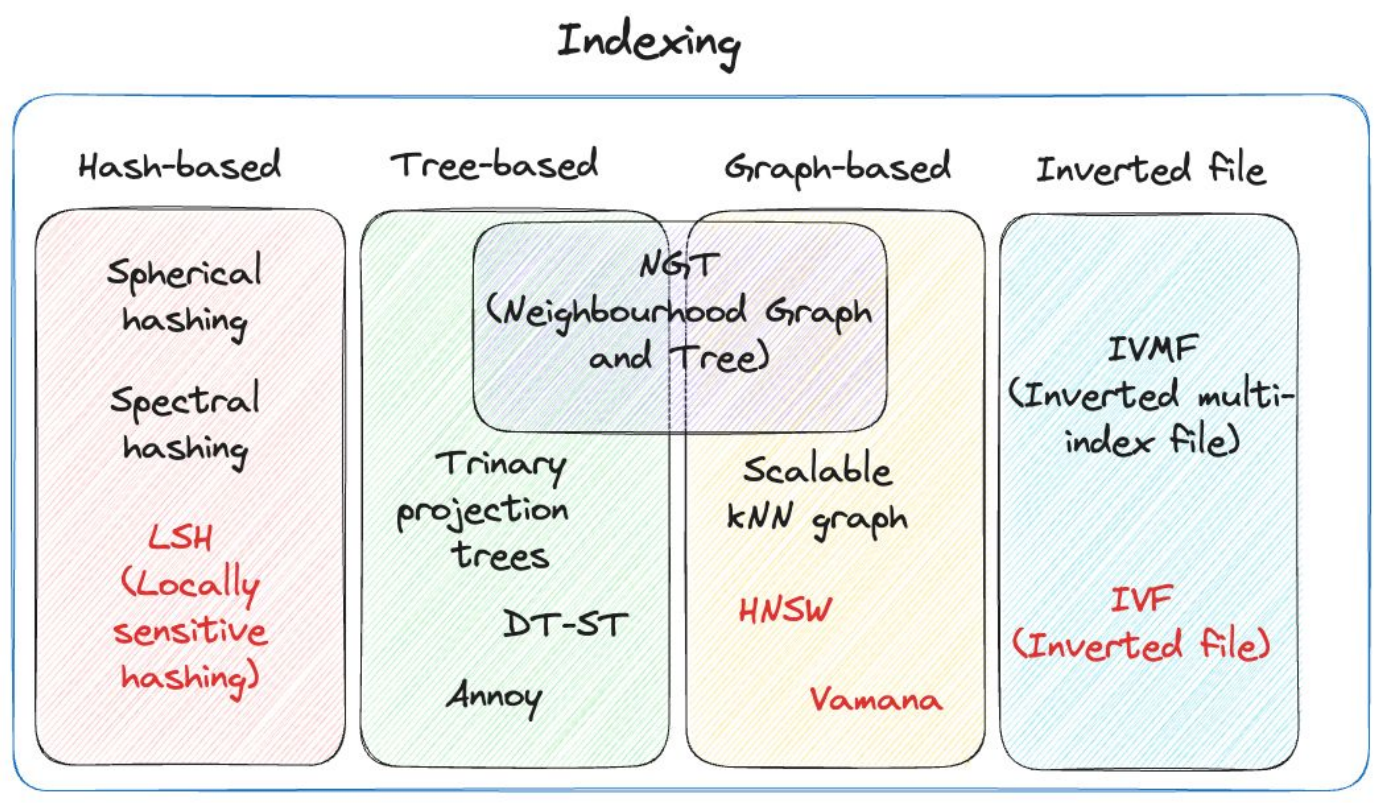
Maybe even some multimodal data 🎩

- Search Quality is declining
- Index updates take forever
- Your frustration rises!

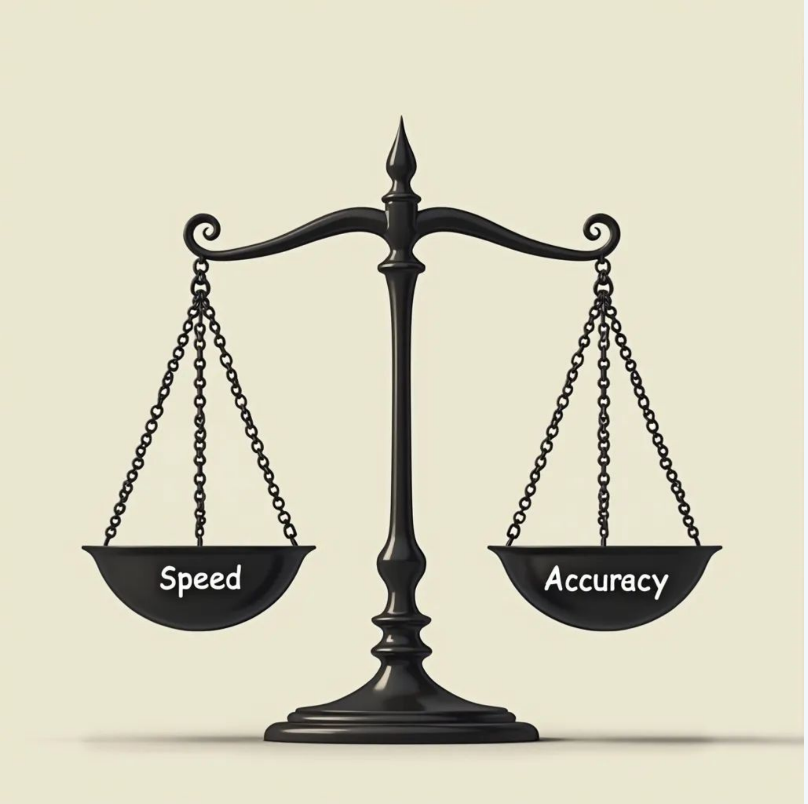
LET THE GAME BEGIN



Index Types Overview

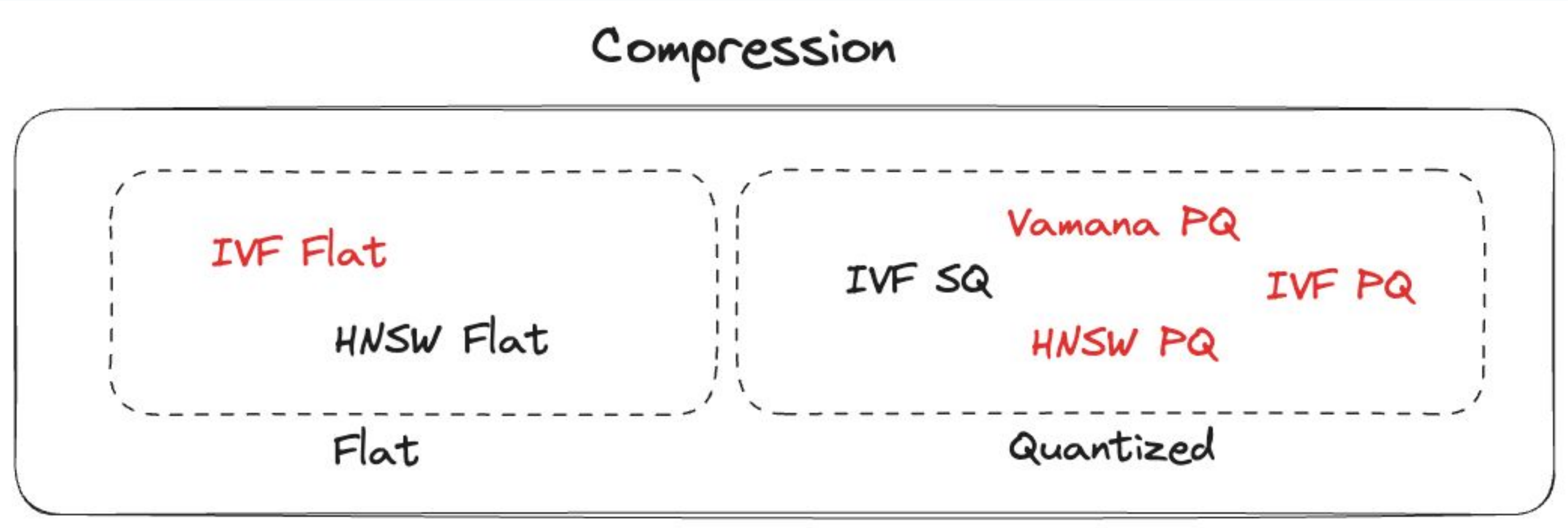


Balancing Speed & Accuracy



zilliz

Compression



zilliz


The Journey Continues

The Distributed Dilemma

✅ You've optimized your Index!

✅ You've chosen the right data structure

✅ You've dabbled in compression techniques!

More data → One instance isn't enough anymore!

⇒ It's time to go **Distributed**!!

Sharding: Divide and Conquer

You brainstorm different sharding approaches:

- **Random sharding**: Simple, but might lead to uneven distribution.
- **Hash-based sharding**: More even distribution, but potentially tricky with updates.
- **Semantic sharding**: Grouping similar vectors together, which could speed up certain types of queries.

Partitioning!

You sketch out a plan:

1. Incoming data gets routed to the appropriate language-specific partition.

2. Queries first determine the relevant language(s).

3. The search is executed only on the relevant partition.

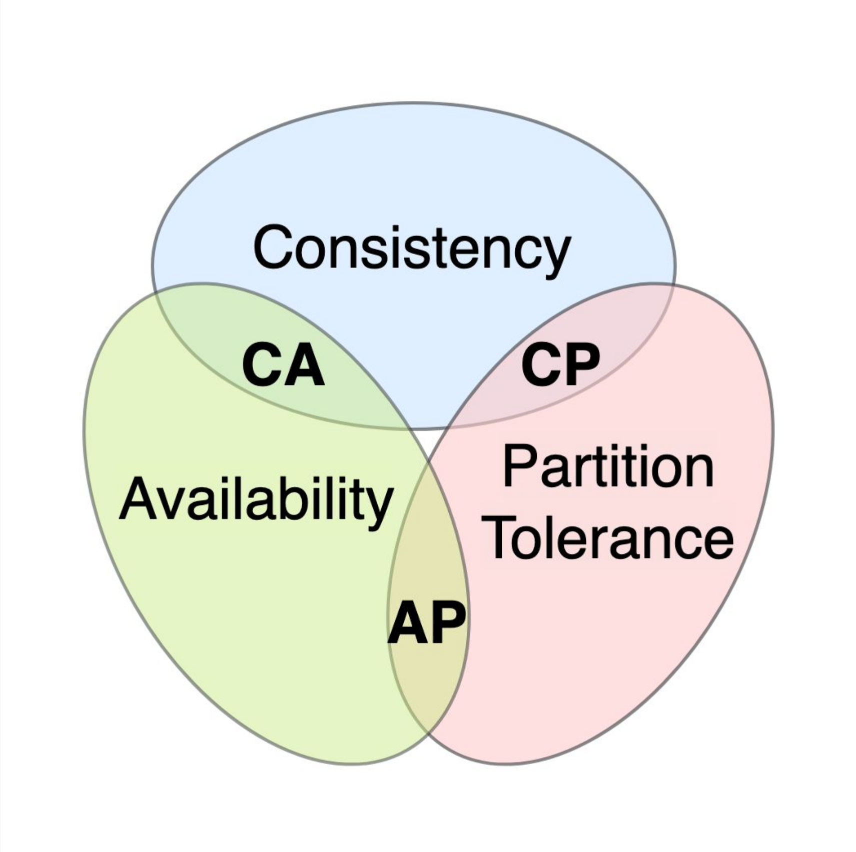
This could reduce the search space for many queries, leading to faster results!

Scaling Horizontally: The Replica Game

- You decide to implement a replica system:
  - Each shard has multiple replicas.
  - Read requests are load-balanced across replicas.
  - Write operations are synchronized across all replicas of a shard.

This setup allows you to scale out your read capacity simply by adding more replicas.

Consistency vs Availability



What about Real-Time?!

Users want it now!

- Write Ahead Logs
- Two-tier Systems
- Incremental Indexing

And Monitoring?! Keeping the Beast in Check

You set up dashboards to track:

- Query latency across different shards and replicas
- Index update times
- Resource utilization (CPU, memory, disk I/O)
- Shard balance and data distribution

You also implement automated processes for:

- Rebalancing shards when they become uneven
- Adding or removing replicas based on load
- Performing rolling updates to minimize downtime

zilliz

To Host or Not to Host?

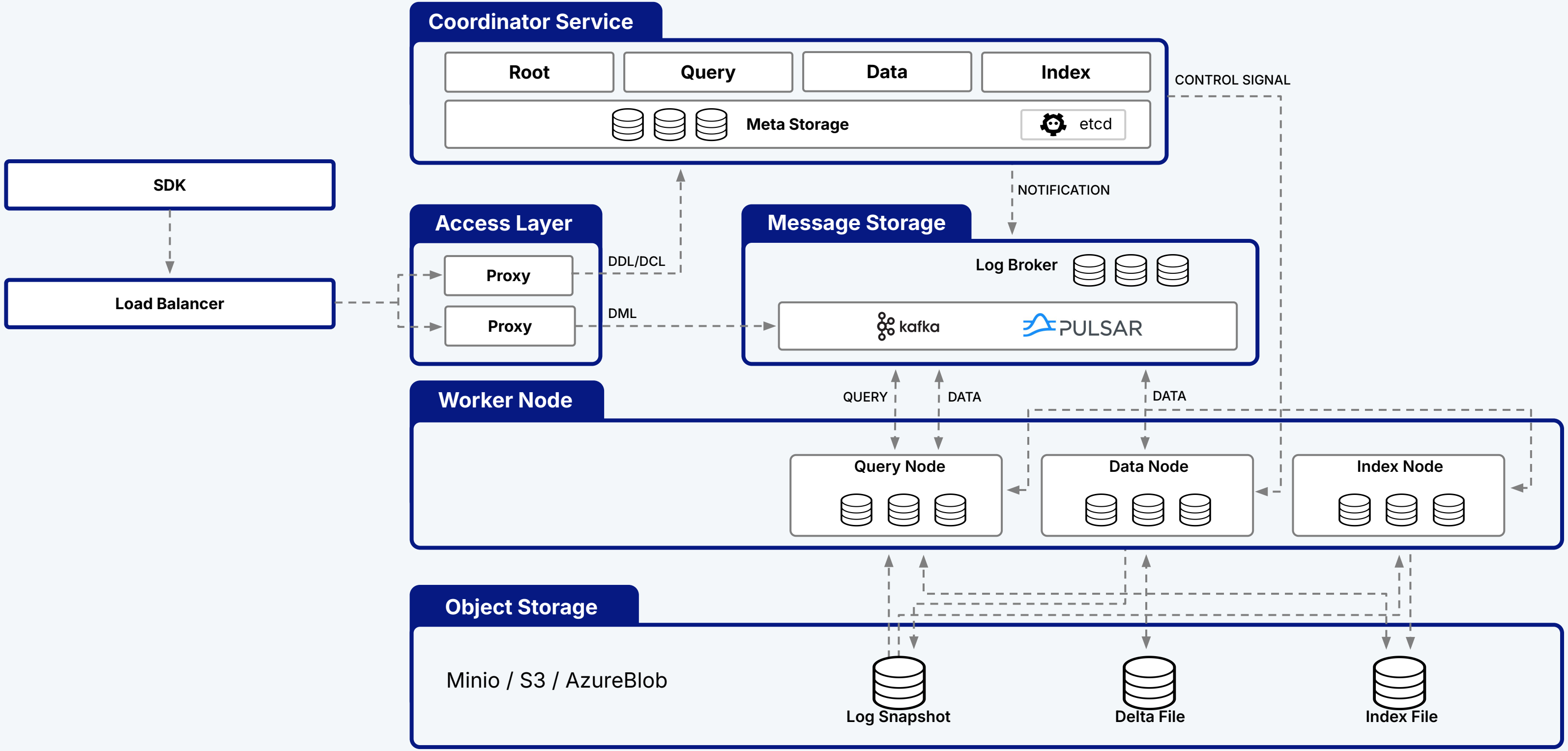
As your system grows, you start weighing the pros and cons of cloud-hosted solutions versus managing your own infrastructure. You consider factors like:

- Scalability and elasticity
- Operational overhead
- Cost predictability
- Data privacy and compliance requirements

A Never Ending Journey

Congratulations, you just built Milvus!

Fully Distributed Architecture



Achieving Billion+ Scale vector Search with K8s

zilliz

Milvus 🤝 Open-Source

MINIO

Store Vectors and Indexes
Enables Milvus' **stateless** architecture

Kafka/ Pulsar

Handles **Data Insertion** stream
Internal Component Communications
Real-time updates to Milvus

Prometheus / Grafana

Collects **metrics** from Milvus
Provides **real-time monitoring** dashboards

Kubernetes

Milvus Operator CRDs

zilliz

Milvus Data Structures

Shard

- Boost the ingestion rate

Segment

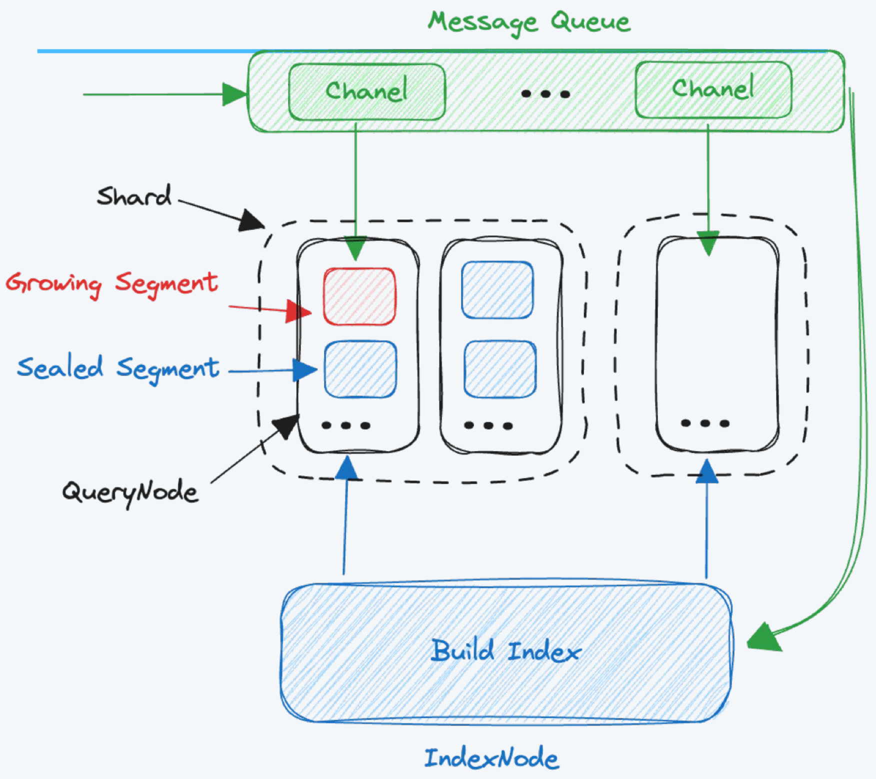
- A single unit of Data in Milvus.
  Segment < Partition < Collection

Growing Segment

- Directly retrieves **data** from the **message queue** for rapid service. Utilizes a **brute-force index** and prioritizes **data freshness**.

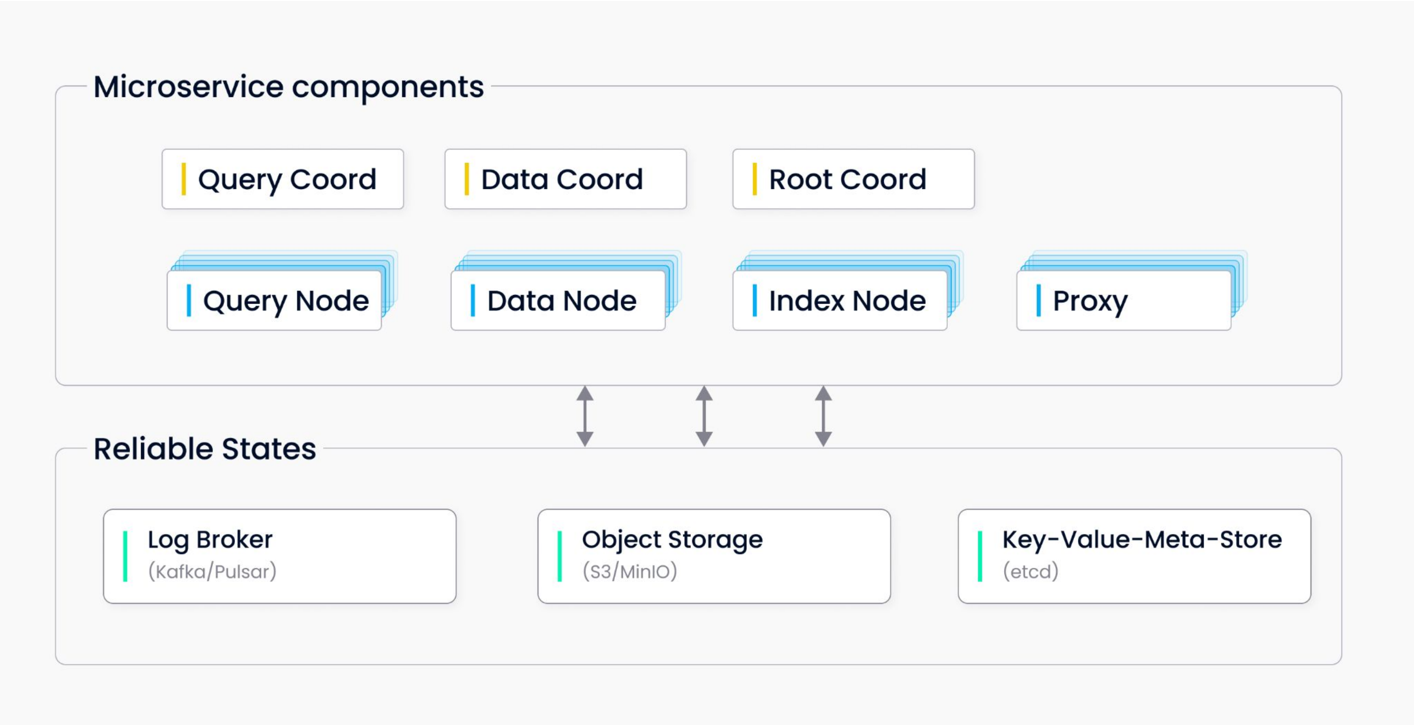
Sealed Segment

- An immutable segment uses **indexing** methods to guarantee **efficiency.**



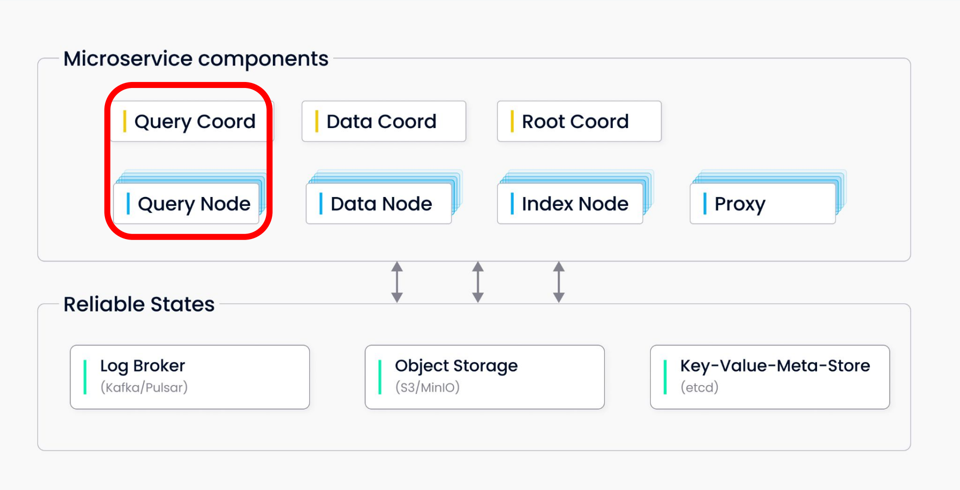
zilliz

Distributed Architecture



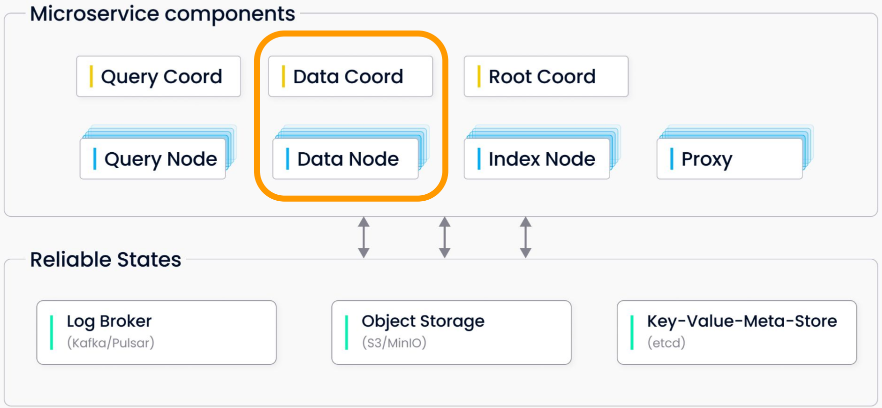
zilliz

Query Node: Serving Search Requests



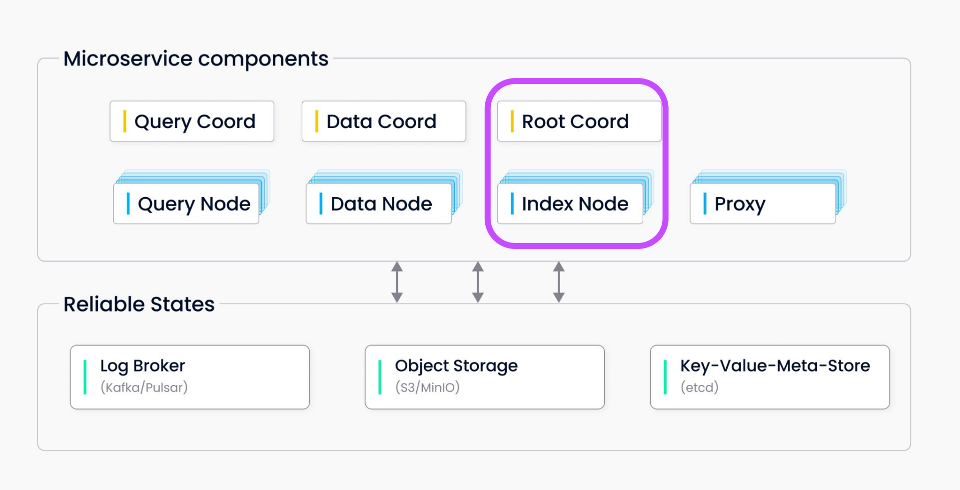
- Subscribe to the **log broker** for real-time querying
- Convert new data into **Growing Segments** - temporary in-memory structures for the latest information.
- Access **Sealed Segments** from object storage for comprehensive searches.
- Perform **hybrid searches** combining vector and scalar data for accurate retrieval.

Data Node: Processing Data Updates



- Subscribe to the **log broker** for real-time updates.
- Process **mutation requests** for data changes or updates.
- Pack log data into **log snapshots -** compressed bundles of updates.
- Store log snapshots in **object storage** for persistence and scalability

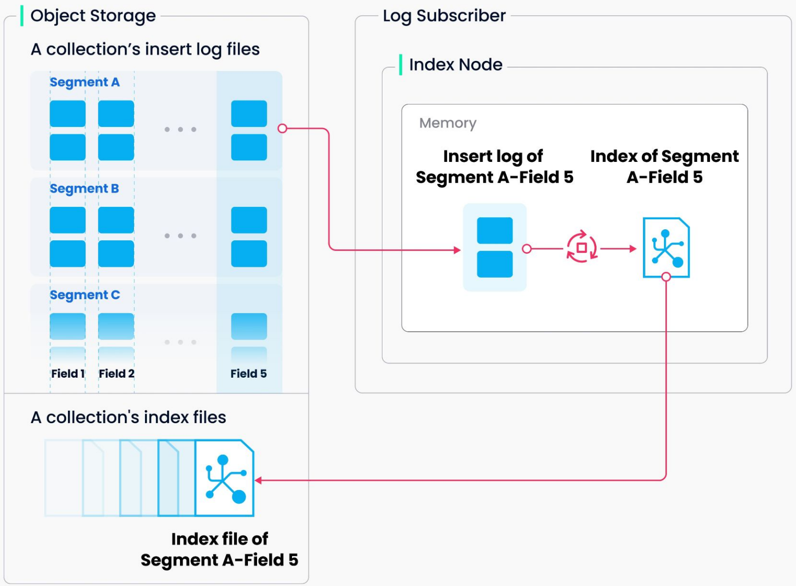
Index Node: Building Search Indexes



- Build indexes on the data to facilitate faster search operations.
- Can be implemented using a **serverless framework** for cost-efficiency and scalability.

zilliz

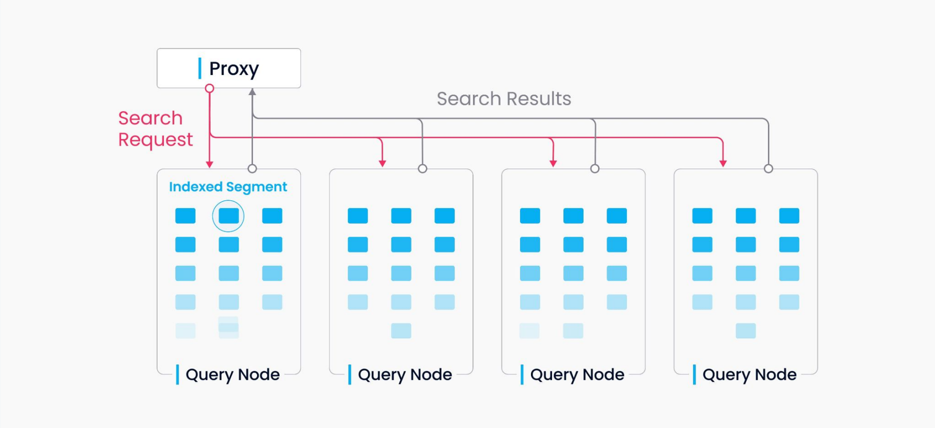
Index Building



To **avoid frequent index building** for data updates.

A collection in Milvus is divided further into segments, each with its own index.

Scalable Search



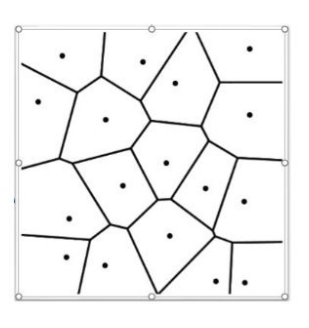
- **Distributed Search** across shards
- **Parallel** Processing
- **Query Optimization**

zilliz

Nearest Neighbors

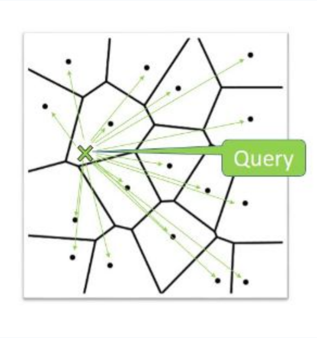
Brute-force

- Exhaustively search query vectors against index dataset



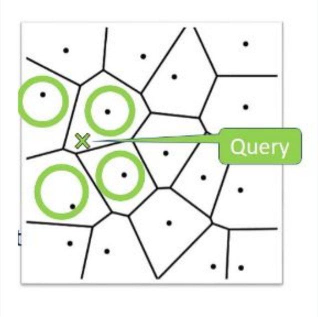
IVF-Flat

- Train clusters on Index dataset
- Partition Vectors by their closest clusters
- Search Smaller number closest lists.



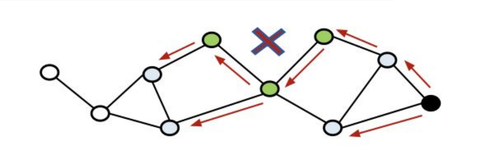
IVF-PQ

- Train clusters on Index dataset
- Partition and compress vectors by their closest clusters
- Search smaller number of closest lists



CAGRA

- Build a graph from vector neighborhoods
- Reduce distances during search by traversing graph



zilliz

CAGRA

GPU accelerated Graph-Based ANN from NVIDIA

- Similar to HNSW for CPU
- Individual queries parallelized during Search
- Higher throughput than usual GPU Graph ANNs and lower latency than CPU Graph ANNs.

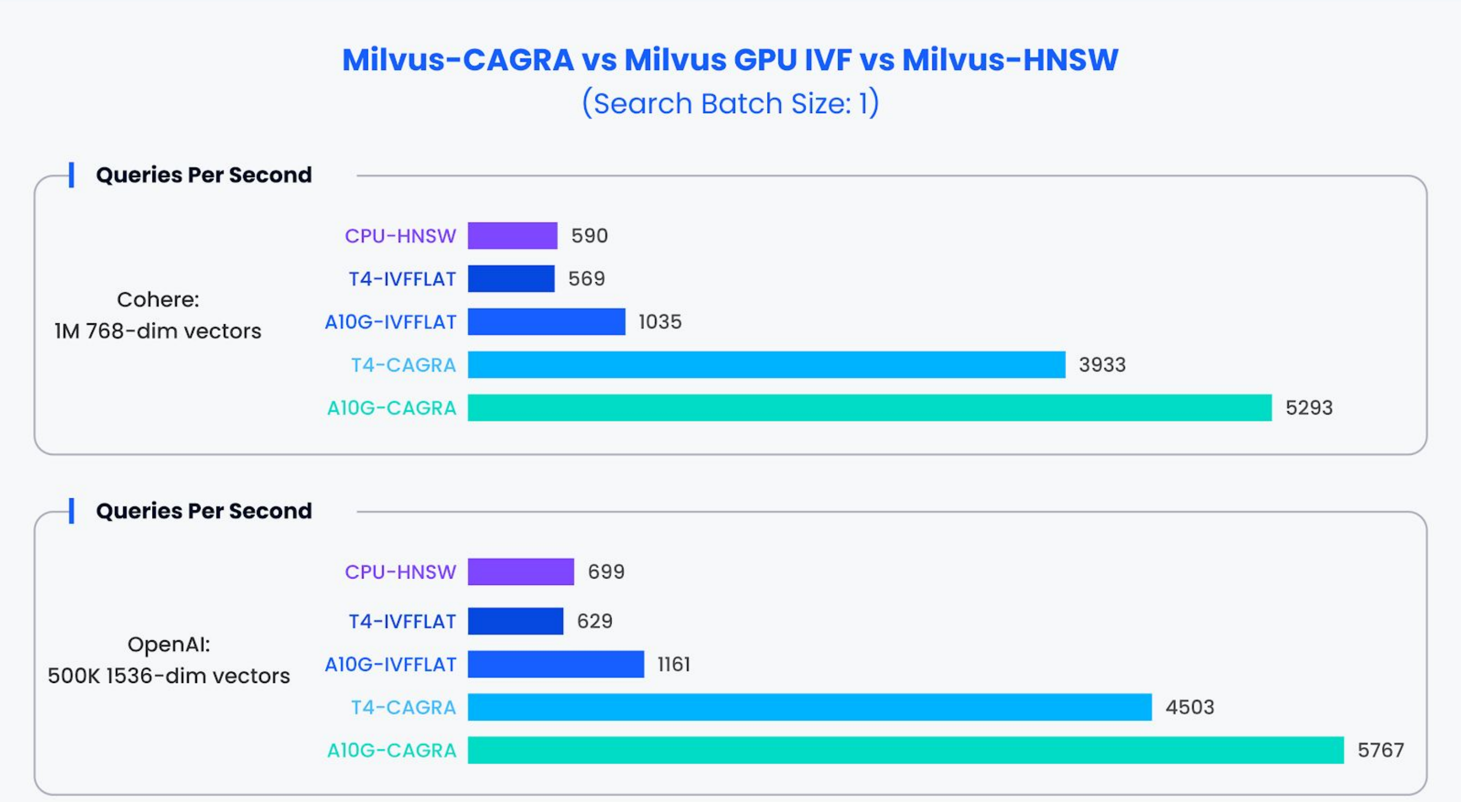
zilliz

Benchmarking setup

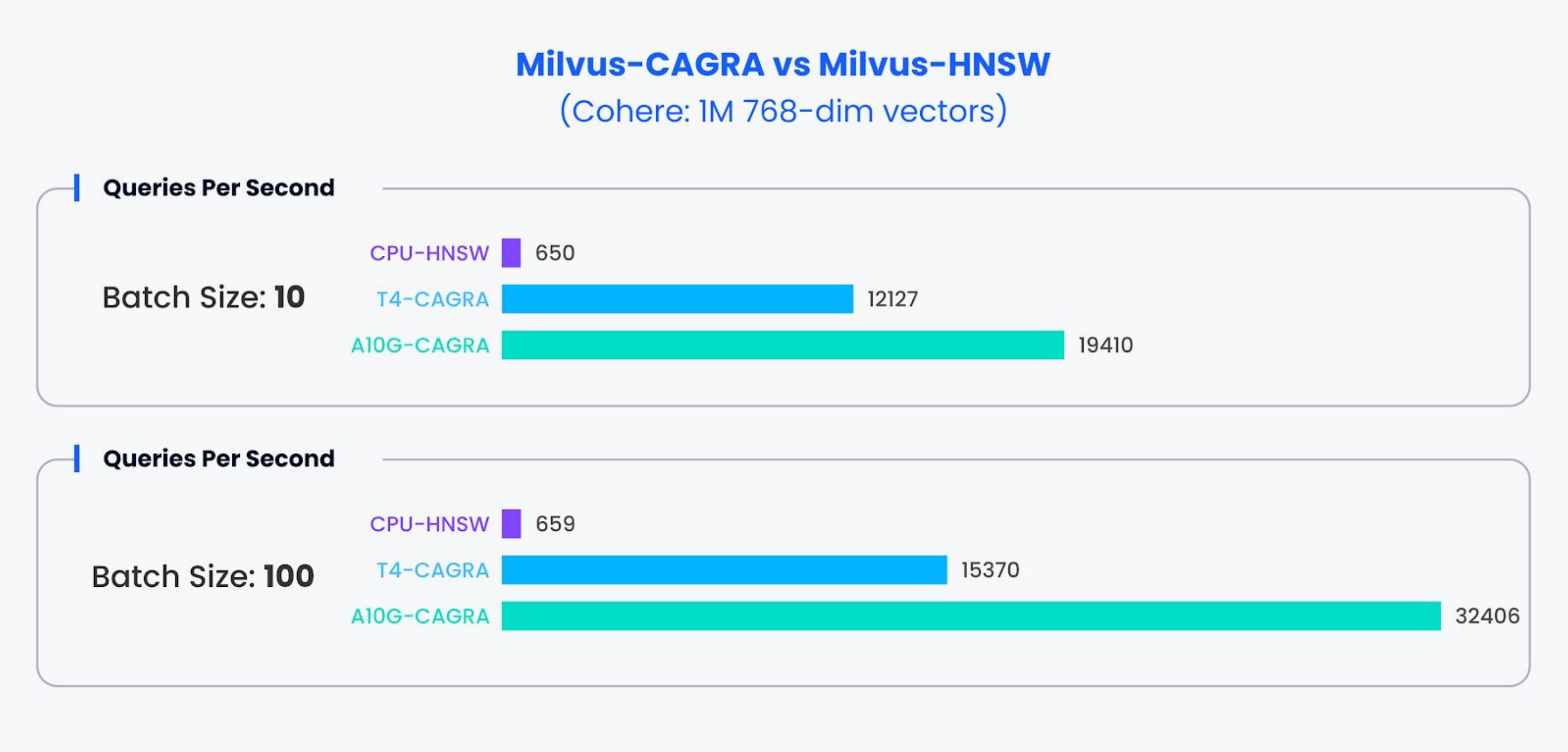
	Instance type	vCPU	Memory	GPU memory	Price($/h)	
T4	g4dn.2xlarge	8	32G	16G	0.752	1.58x expensive
A10G	g5.2xlarge	8	32G	24G	1.212	2.55x expensive
CPU	m6id.2xlarge	8	32G	–	0.4746	

zilliz

Small Batch Performance

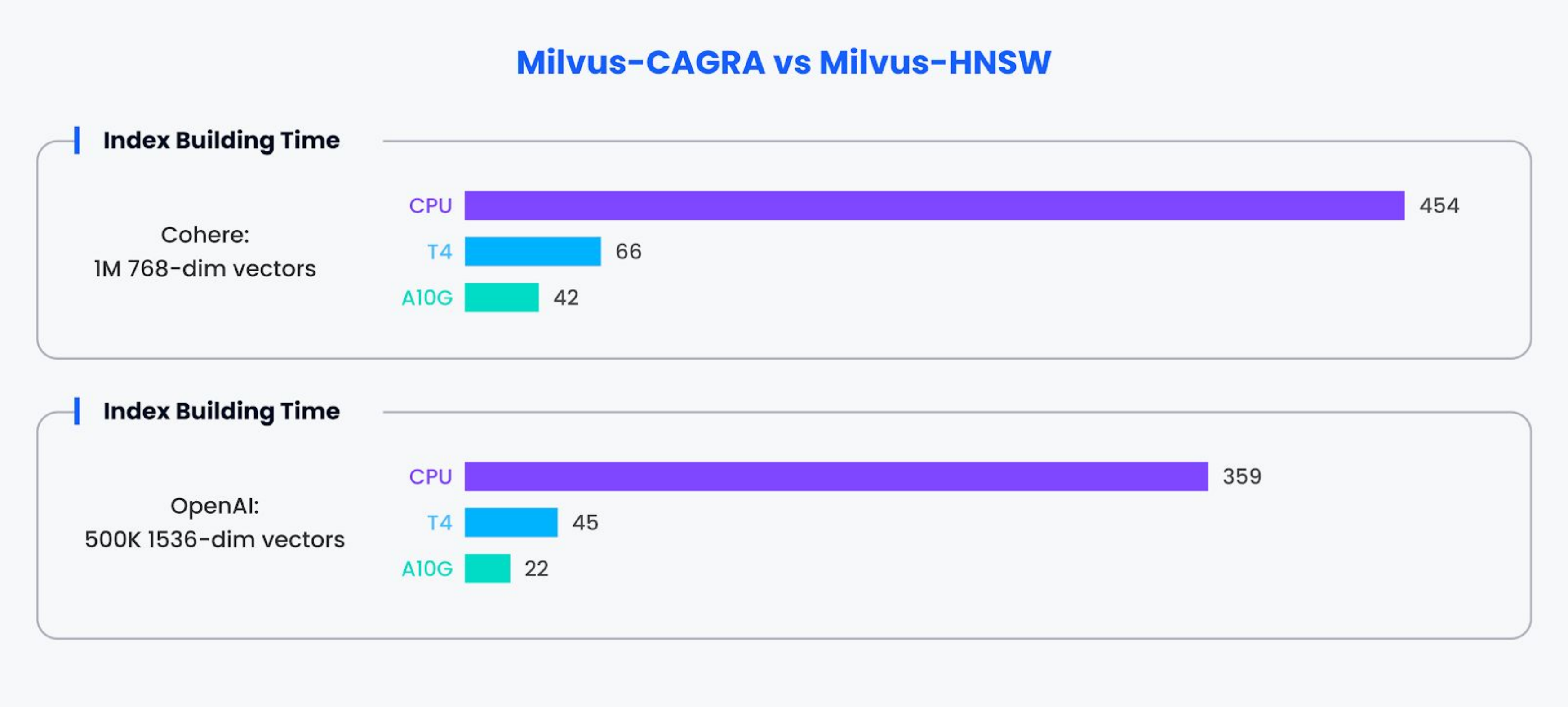


Large Batch Performance



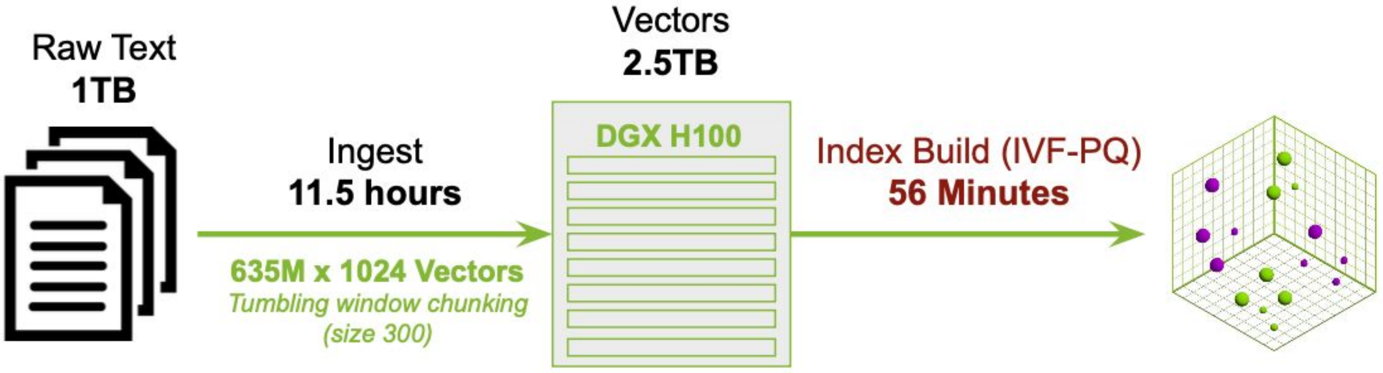
 |

Index Building (in seconds)



Scale Indexing on DGX H100 (8x H100)

GPU
Benchmark



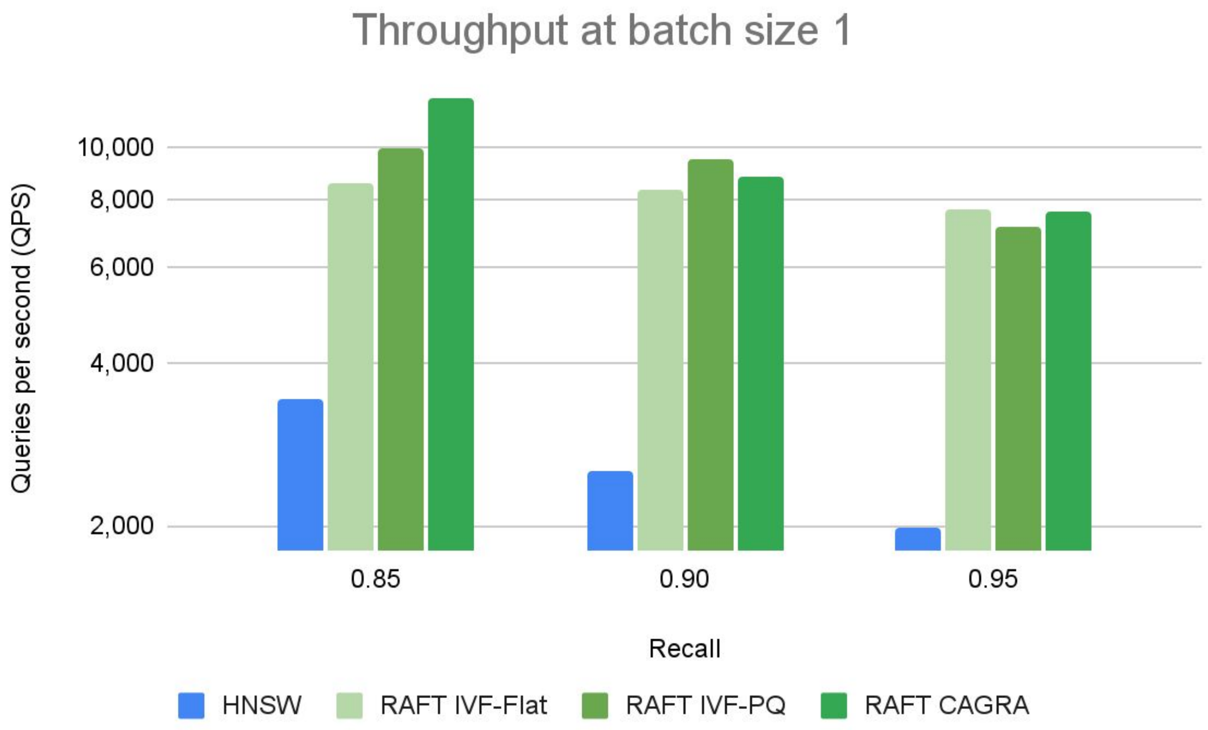
CPU
Estimate

GPU Index Build Time	56 Min
Number of GPUs	× 8
GPU / CPU Performance	× 20
Minutes per day	÷ 1440 Min/Day
CPU Estimated Index Build	**6.22 Days**

linear speedup

zilliz

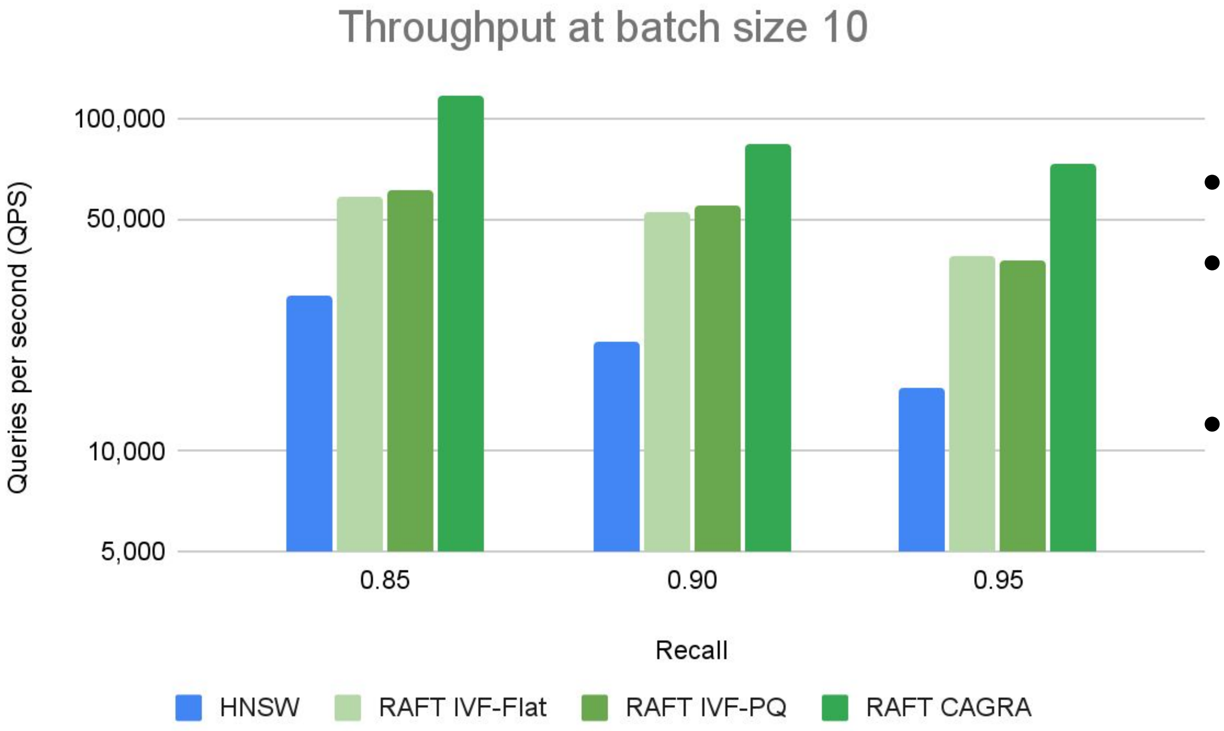
Benchmarks (Batch size 1)



- **DEEP-100M** dataset
- **H100** GPU for RAFT indexes
- Intel **Xeon Platinum** 8480CL CPU for HNSW

zilliz

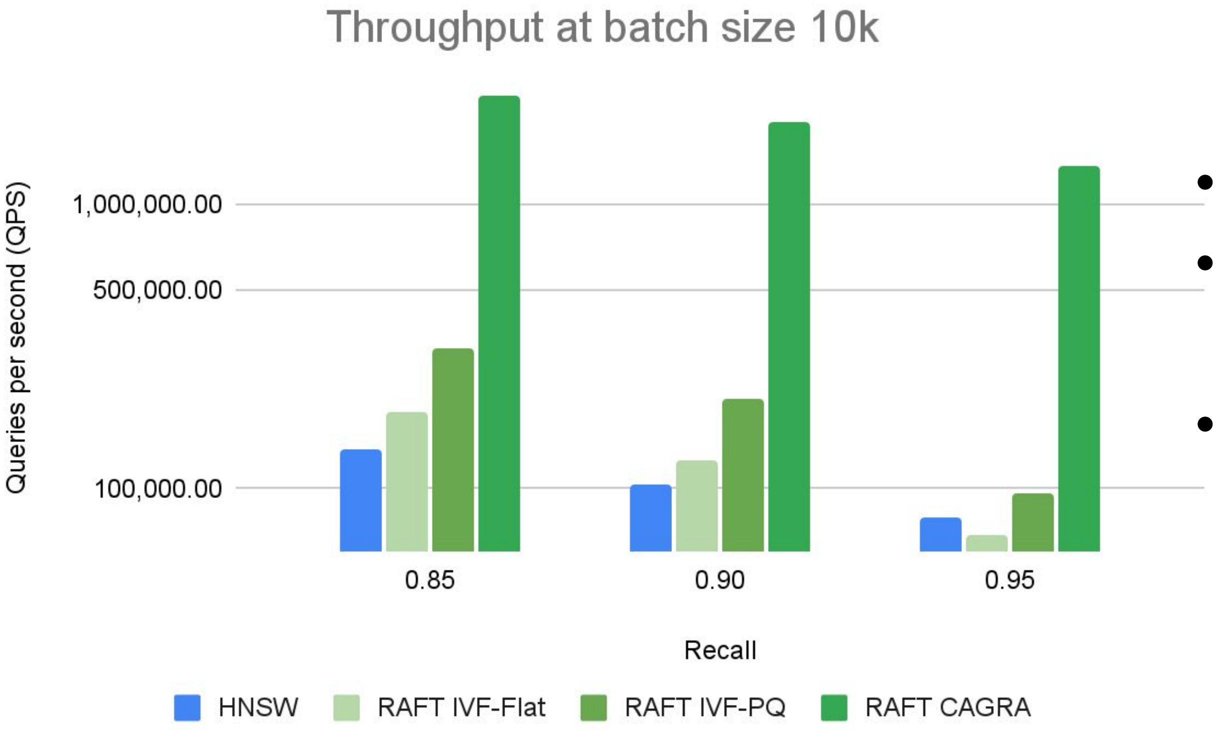
Benchmarks (Batch size 10)



- **DEEP-100M** dataset
- **H100** GPU for RAFT indexes
- Intel **Xeon Platinum** 8480CL CPU for HNSW

zilliz

Benchmarks (Batch size 10K)



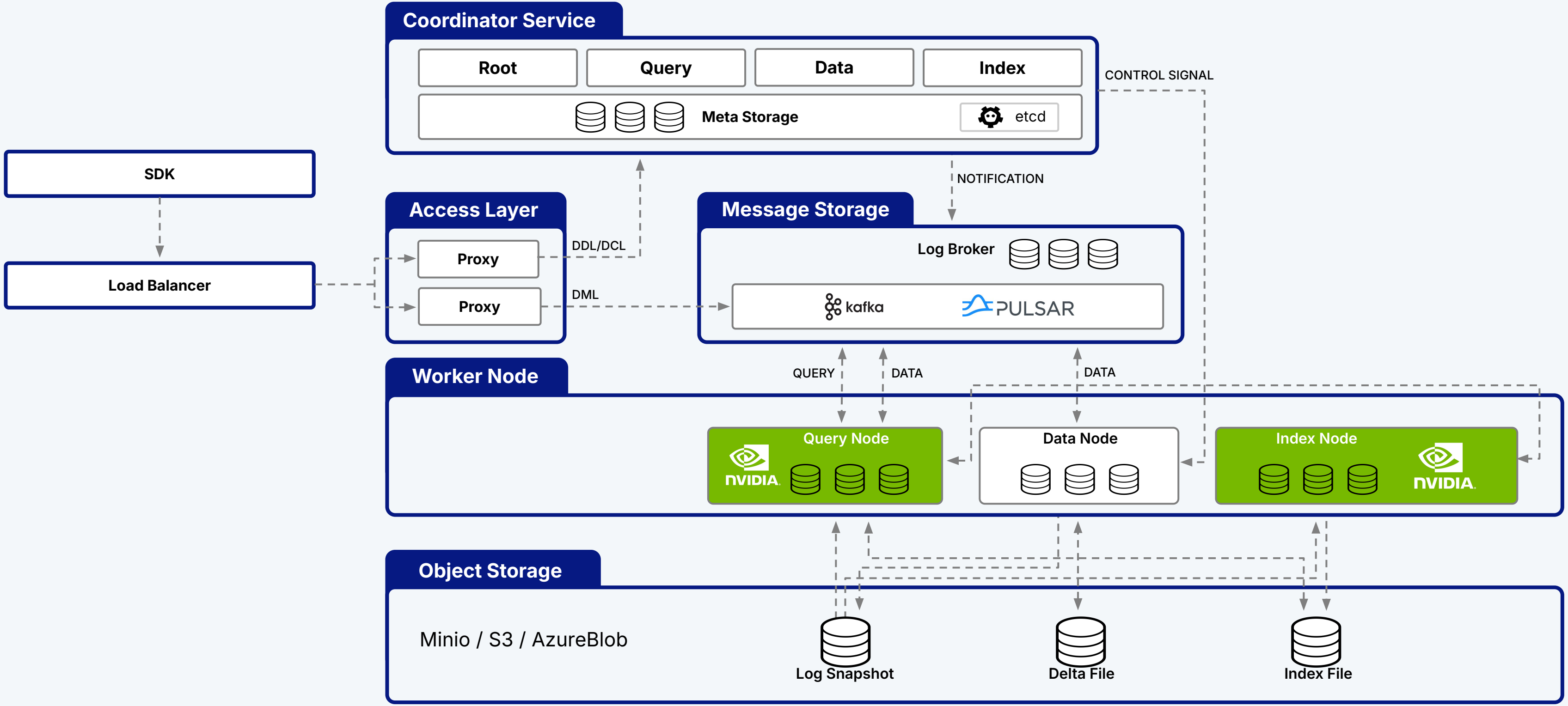
- **DEEP-100M** dataset
- **H100** GPU for RAFT indexes
- Intel **Xeon Platinum** 8480CL CPU for HNSW

Tips for more performance

- Increase the number of **Data Nodes** ⇒ Improve **Streaming ingestion** performance
- Place **Index** and **Query nodes** on **separate GPUs**
- **Increase** max **segment** size can **improve search latency and throughput**

zilliz

Fully Distributed Architecture



But at what Scale?

10B vectors

of 1536 dimensions
in a single Milvus/Zilliz Cloud instance

100B vectors

in one of the largest deployment running on K8s.

Speaker

Stephen Batifol

Developer Advocate, Zilliz/ Milvus

stephen.batifol@zilliz.com
linkedin.com/in/stephen-batifol/
@stephenbtl

Thank you

milvus.io

github.com/milvus-io/

@milvusio

Notebooks

@stephenbtl

/in/stephen-batifol

zilliz

Ready to scale 🚀

Write your code **once**, and run it **everywhere**, at **scale**!

- API and SDK are the same

Milvus Lite

- Ideal for **prototyping**, **small scale** experiments.
- Easy to set up and use, `pip instally pymilvus`
- Scale to ≈**1M** vectors

Milvus Standalone

- Single-Node Deployment
- Bundled in a **single Docker** Image
- Supports Primary/ Secondary
- Scale up to **100M vectors**

Milvus Distributed

- Run on **K8s**
- **Load balancer** and **Multi-Node** Management
- **Scaling** of each component **independently**
- Scale to **100B** vectors

zilliz

Well-connected in LLM infrastructure to enable RAG use cases

LangChain | Semantic Kernel | ***Framework*** | LlamaIndex | haystack by deepset

Software Infrastructure

Embedding Models

PyTorch | Hugging Face | cohere | jina

Vector Database

milvus | zilliz

LLMs

OpenAI | Inflection | ANTHROPIC | LLaMA by Meta

NVIDIA | intel | ***Hardware Infrastructure*** | aws | Google Cloud | Azure

zilliz

Thank you

milvus.io

github.com/milvus-io/

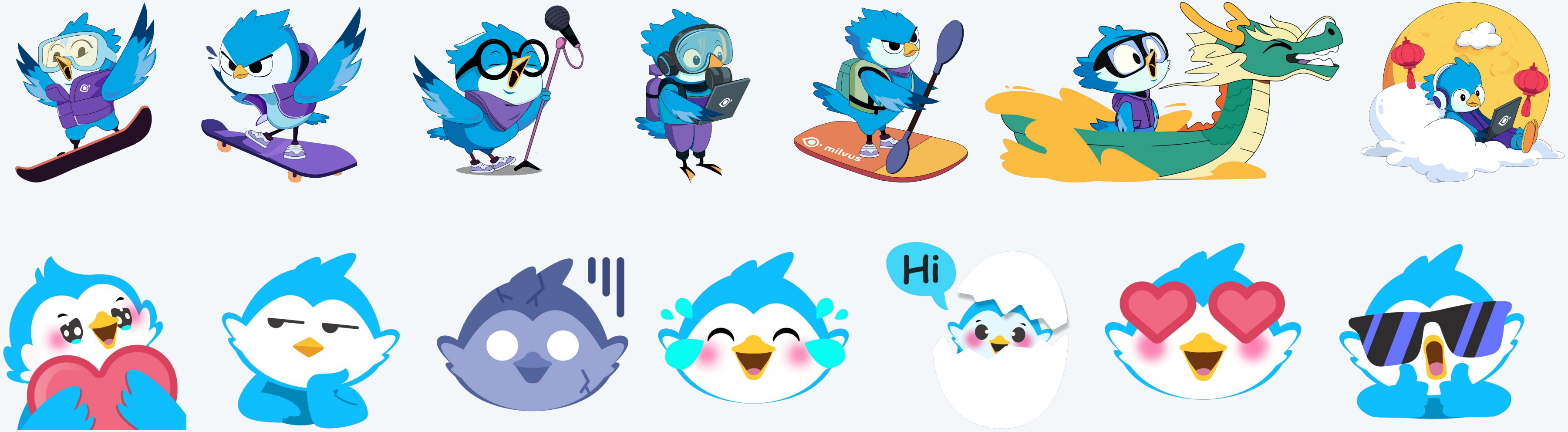
@milvusio

Notebooks

@stephenbtl

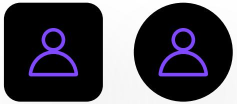
/in/stephen-batifol

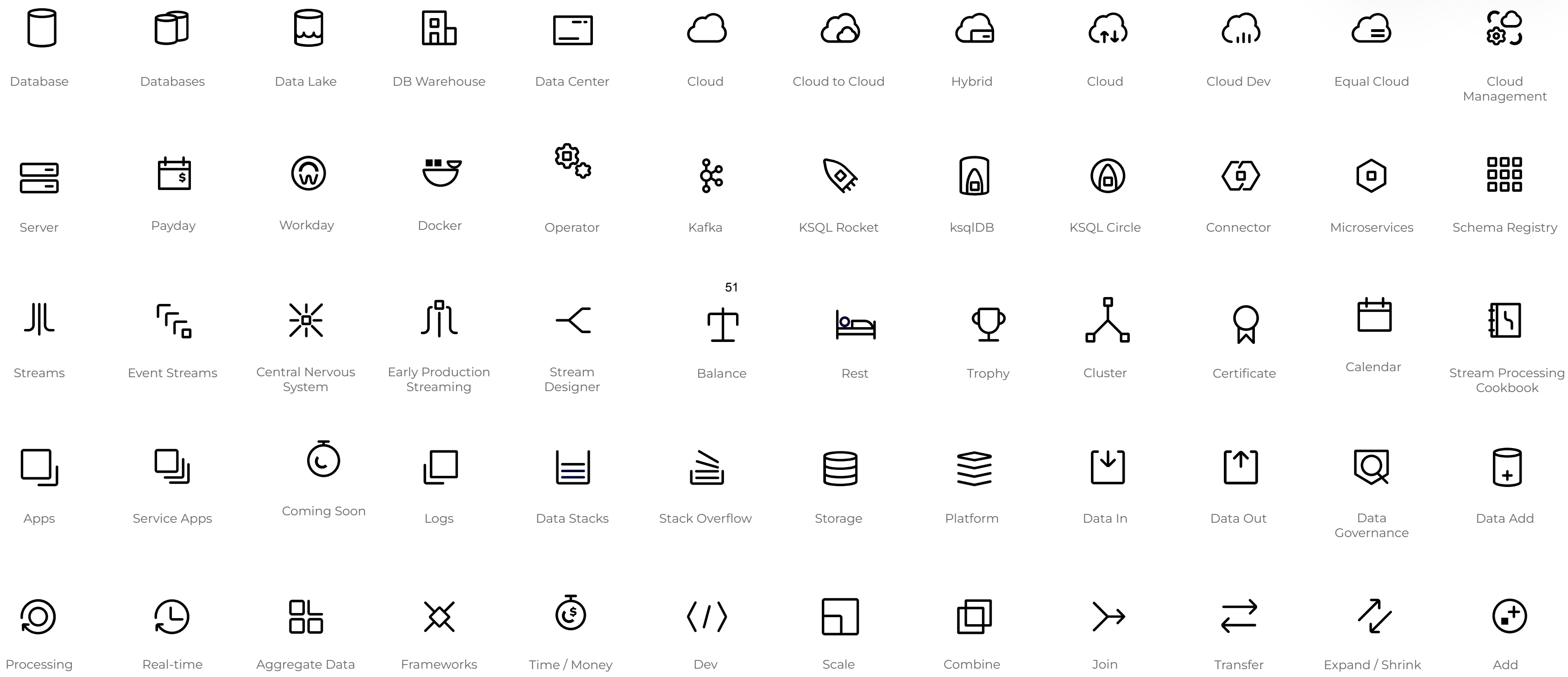
zilliz


milvus
Hi

Icons

ICON STYLE:





For the complete, most updated collection of Icons please go to: **https://cnfl.io/Icons**

Icons

ICON STYLE:

Globe	Infinity	Settings	Monitoring	Anomaly Detection	Analytics	Real-time Analytics	Real-time	Processing	Process Data	Upload	Download
Computer	Devices	Computer / DB / Cloud	Speed	Time	Web Confirmed	RSS	ROI	Message	Quotes	Interview	# of Topics
Person	People	Webinar	Developer	Onboard	Offboard	People Manager	Career Enablement	Roadmap	Filter	Search	Solution
Features	Company Policies	Docs	Invoice	Blog	Podcast	Video	Book	Table	Email	Question	Check
Lock	Key	Warning	Hacker	Bug	GDPR	CCPA	Shield	Shield Open	Machine Learning	Continuous Learning	Eye

52

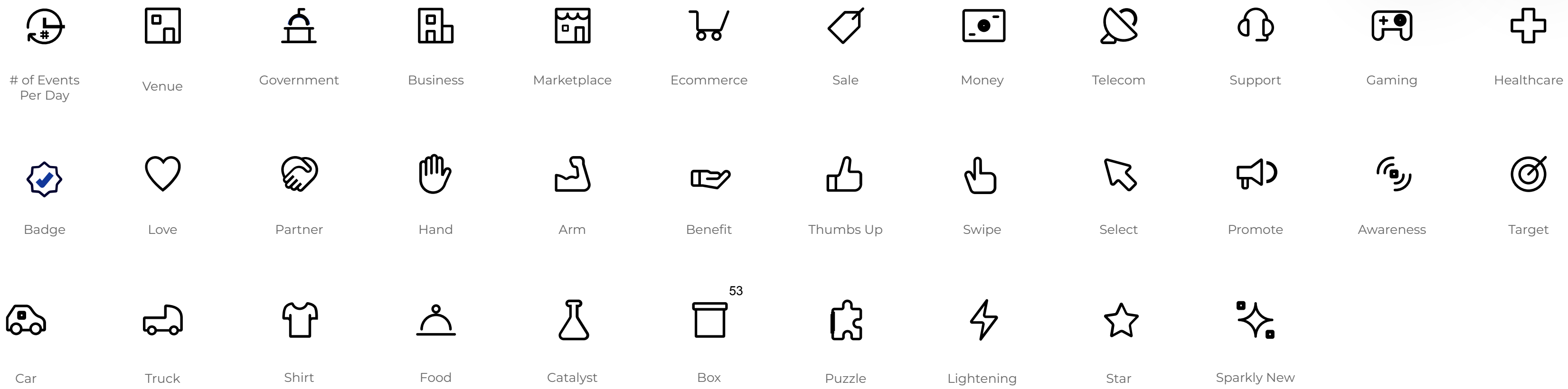
Current

For the complete, most updated collection of Icons please go to: **https://cnfl.io/Icons**

zilliz

Icons

ICON STYLE:



Current

For the complete, most updated collection of Icons please go to: **https://cnfl.io/Icons**

zilliz