id: comparison.md
title: What is New in Milvus 2.0

What is New in Milvus 2.0

We recommend you trying out Milvus 2.0. Here is why:

Design concepts

As our next-generation cloud-native vector database, Milvus 2.0 is built around the following three principles:

Cloud-native first: We believe that only architectures supporting storage and computing separation can scale on demand and take the full advantage of cloud's elasticity. We'd also like to bring your attention to the microservice design of Milvus 2.0, which features read and write separation, incremental and historical data separation, and CPU-intensive, memory-intensive, and IO-intensive task separation. Microservices help optimize allocation of resources for the ever-changing heterogeneous workload.

Logs as data: In Milvus 2.0, the log broker serves as the system' backbone: All data insert and update operations must go through the log broker, and worker nodes execute CRUD operations by subscribing to and consuming logs. This design reduces system complexity by moving core functions such as data persistence and flashback down to the storage layer, and log pub-sub make the system even more flexible and better positioned for future scaling.

Unified batch and stream processing: Milvus 2.0 implements the unified Lambda architecture, which integrates the processing of the incremental and historical data. Compared with the Kappa architecture, Milvus 2.0 introduces log backfill, which stores log snapshots and indexes in the object storage to improve failure recovery efficiency and query performance. To break unbounded (stream) data down into bounded windows, Milvus embraces a new watermark mechanism, which slices the stream data into multiple message packs according to write time or event time, and maintains a timeline for users to query by time.

Product highlights

The costs of running a database involve not only runtime resource consumption, but also the potential learning costs and the operational and maintenance costs. Practically speaking, the more user-friendly a database is, the more likely it is going to save such potential costs. From Milvus' calendar day one, ease of use is always put on the top of our list, and the latest Milvus 2.0 has quite a few to offer in the way of reducing such costs.

Always online

Data reliability and service sustainability are the basic requirements for a database, and our strategy is "fail cheap, fail small, and fail often".

- "Fail cheap" refers to storage and computing separation, which makes the handling of node failure recovery straightforward and at a low cost.
- "Fail small" refers to the "divide and conquer" strategy, which simplifies the design complexity by having each coordinator service handle only a small portion of read/write/incremental/historical data.
- "Fail often" refers to the introduction of chaos testing, which uses fault injection in a testing environment to simulate situations such as hardware failures and dependency failures and accelerate bug discovery.

Hybrid search between scalar and vector data

To leverage synergy between structured and unstructured data, Milvus 2.0 supports both scalar and vector data and enables hybrid search between them. Hybrid search helps users find the approximate nearest neighbors that match a filter criteria. Currently, Milvus supports relational operations such as EQUAL, GREATER THAN, and LESS THAN, and logical operations such as NOT, AND, OR, and IN.

Tunable consistency

As a distributed database abiding by the PACELC theorem, Milvus 2.0 has to trade off between consistency and availability & latency. In most scenarios, overemphasizing data consistency in production can overkill because allowing a small portion of data to be invisible has little impact on the overall recall but can significantly improve the query performance. Still, we believe that consistency levels, such as strong, bounded staleness, and session, have their own unique application. Therefore, Milvus supports tunable consistency at the request level. Taking testing as an example, users may require strong consistence to ensure test results are absolutely correct.

Time travel

Data engineers often need to do data rollback to fix dirty data and code bugs. Traditional databases usually implement data rollback through snapshots or even data retrain. This could bring excessive overhead and maintenance costs. Milvus maintains a timeline for all data insert and delete operations, and users can specify a timestamp in a query to retrieve a data view at a specified point in time. With time travel, Milvus can also implement a lightweight data backup or data clone.

ORM Python SDK

Object relational mapping (ORM) allows users to focus more on the upper-level business model than on the underlying data model, making it easier for developers to manage relations between collections, fields, and programs. To close the gap between proof of concept (PoC) for AI algorithms and production deployment, we engineered PyMilvus ORM APIs, which can work with an embedded library, a standalone deployment, a distributed cluster , or even a cloud service. With a unified set of APIs, we provide users with a consistent user experience and reduce code migration or adaptation costs.

ORM_Python_SDK

Support tools

- **Milvus Insight** is Milvus's graphical user interface offering practical functionalities such as cluster state management, meta management, and data query. The source code of Milvus Insight will also be open sourced as an independent project. We are looking for more contributors to join this effort.
- **Out-of-box experience (OOBE), faster deployment:** Milvus 2.0 can be deployed using helm or Docker Compose.
- Milvus 2.0 uses Prometheus, an open-source time-series database, to store performance and monitor data, and Grafana, an open observability platform, for metrics visualization.

Milvus 2.0 vs. 1.x: Cloud-native, distributed architecture, highly scalable, and more

	Milvus 2.0	**Milvus 1.x**
Architecture	Cloud native	Shared storage
Scalability	500+ nodes	1 to 32 read nodes with only one write node
Durability	• Object storage service (OSS) • Distributed file system (DFS)	• Local disk • Network file system (NFS)
Availability	99.9%	99%
Data consistency	Three levels of consistency: • Strong • Bounded Staleness • Session • Consistent prefix	Eventual consistency
Data types supported	• Vectors • Fixed-length scalars • String and text (in planning)	Vectors

	Milvus 2.0	Milvus 1.x
Basic operations supported	• Data insertion • Data deletion (in planning) • Data query • Approximate nearest neighbor (ANN) Search • Recurrent neural network (RNN) search (in planning)	• Data insertion • Data deletion • Approximate nearest neighbor (ANN) Search
Advanced features	• Scalar filtering • Time Travel • Multi-site deployment and multi-cloud integration • Data management tools	• Mishards • Milvus DM
Index types and libraries	• Faiss • Annoy • Hnswlib • RNSG • ScaNN (in planning) • On-disk index (in planning)	• Faiss • Annoy • Hnswlib • RNSG
SDKs	• Python • Go (in planning) • Java (in planning) • RESTful (in planning) • C++ (in planning)	• Python • Java • Go • RESTful • C++
Release status	Release candidate. A stable version will be released in August.	Long-term support (LTS)

---

id: milvus_adopters.md
title: Milvus Adopters

Milvus Adopters

Milvus is trusted by over 1,000 organizations worldwide and has applications in almost every industry. What follows is a list of Milvus users that have adopted the software in production.

Company	Industry	User Story
Beike	Real estate	Making With Milvus: AI-Infused Proptech for Personalized Real Estate Search
DXY.cn	Medical	
Deepset.ai	Software	Semantic Search with Milvus, Knowledge Graph QA, Web Crawlers and more!
EnjoyMusic Technology	Entertainment	Item-based Collaborative Filtering for Music Recommender System
Getir	Instant Grocery	
Hewlett-Packard (HP)	Information technology	
Juicedata	Software	
Kingsoft	Software	Building an AI-Powered Writing Assistant for WPS Office
Lucidworks	Software	Build Semantic Search at Speed
Meetsocial Group	Marketing	
Miao Health	Health technology	
Mozat	Online social	
Opera	Software	
Sohu	Internet	Building an Intelligent News Recommendation System Inside Sohu News App
The Cleveland Museum of Art	Arts	ArtLens AI: Share Your View

Company	Industry	User Story
Tokopedia	Online Retail	How we used semantic search to make our search 10x smarter
TrendMicro	Software	Making with Milvus: Detecting Android Viruses in Real Time for Trend Micro
Ufoto	Software	
Vipshop	Online Retail	
Vova	Online Retail	Building a Search by Image Shopping Experience with VOVA and Milvus
Xiaomi	Consumer electronics	Making with Milvus: AI-Powered News Recommendation Inside Xiaomi's Mobile Browser
iYUNDONG	Sports	Extracting Event Highlights Using iYUNDONG Sports App

---

id: overview.md
title: What is Milvus
related_key: Milvus Overview

What is Milvus

Milvus is an open-source vector database built to power AI applications and vector similarity search.

It is available in:

- Milvus standalone
- Milvus cluster

Compatibility:

Milvus version	Python SDK version	Java SDK version	Go SDK version
2.0.0-RC2	2.0.0rc2	Coming soon	Coming soon

Milvus 2.0.0-RC2 is the preview version of 2.0.0. It introduces Golang as the distributed layer development language and a new cloud-native distributed design. The latter brings significant improvements to scalability, elasticity, and functionality.

Overall Architecture

Milvus 2.0 is a cloud-native vector database with storage and computation separated by design. All components in this refactored version of Milvus are stateless to enhance elasticity and flexibility.

The system breaks down into four levels:

- Access layer
- Coordinator service
- Worker nodes
- Storage

Access layer: The interface. The access layer is the front layer of the system and endpoint to users. It is in charge of forwarding requests and gathering results.

Coordinator service: The coordinator service assigns tasks to the worker nodes and functions as the system's brain. There are four coordinator types: root coordinator (root coord), data coordinator (data coord), query coordinator (query coord), and index coordinator (index coord).

Worker nodes: The arms and legs. Worker nodes are dumb executors that follow the instructions from the coordinator service and respond to the read/write requests from the access layer. There are three types of worker nodes: data nodes, query nodes, and index nodes.

Storage: The bones. Storage has three types: meta storage, log broker, and object storage.

- Implemented by etcd, meta storage is used to store metadata such as collection and checkpoint for the coordinator service.
- Implemented by Pulsar, log broker is used mainly to store incremental logs and implement reliable asynchronous notifications.
- Implemented by MinIO or S3, object storage is used mainly to store log snapshots and index files.

Architecture

For more information, see Architecture Overview.

Main Components

A Milvus standalone includes three components:

- Milvus
- etcd
- MinIO

A Milvus cluster includes eight microservice components and three third-party infrastructure service components. Microservice components:

- Root coord
- Proxy
- Query coord
- Query node
- Index coord
- Index node
- Data coord
- Data node

Third-party infrastructure components:

- etcd
- MinIO
- Pulsar

Key features

Millisecond search on trillion vector datasets

Average latency measured in milliseconds on trillion vector datasets.

Simplified unstructured data management

- Rich APIs designed for data science workflows.
- Consistent user experience across laptop, local cluster, and cloud.
- Embed real-time search and analytics into virtually any application.

Reliable, always on vector database

Milvus' built-in replication and failover/failback features ensure data and applications can maintain business continuity in the event of a disruption.

Highly scalable and elastic

Component-level scalability makes it possible to scale up and down on demand. Milvus can autoscale at a component level according to the load type, making resource scheduling much more efficient.

Hybrid search

In addition to vectors, Milvus supports data types such as boolean, integers, floating-point numbers, and more. A collection in Milvus can hold multiple fields for accommodating different data features or properties. Milvus pairs scalar filtering with powerful vector similarity search to offer a modern, flexible platform for analyzing unstructured data.

Unified Lambda structure

Milvus combines stream and batch processing for data storage to balance timeliness and efficiency. Its unified interface makes vector similarity search a breeze.

Community supported, industry recognized

With over 1,000 enterprise users, 6,000+ stars on GitHub, and an active open-source community, you’re not alone when you use Milvus. As a graduate project under the LF AI & Data Foundation, Milvus has institutional support.

Scenarios

Milvus can be used in a wide variety of scenarios spanning artificial intelligence, deep learning, traditional vector calculations, and more. See Milvus Adopters for a list of specific use cases.

Biopharmaceutical/Healthcare

Virtual drug screening, virus structure analysis, protein property prediction, drug polymorph prediction, intelligent medical diagnosis, pathological analysis, and high-precision image retrieval.

E-commerce

Image search, intelligent customer service, product search, product matching, product de-duplicate, personalised recommendation, content recommendation, intelligent QA bots.

Internet Services and More

Personalized music recommendation, real estate listing search and recommendation, intelligent customer service, web search, app store search, text similarity search/news recommendation, removing duplicate video content, video search, and searching commodities by image.

Computer Software and Hardware

Corpus/image analysis and recommendation, intelligent product design.

Advertising, Industrial Design, and Manufacturing

Intelligent poster design, targeted advertising, and product inventory management.

Key Concepts

Unstructured data

Unstructured data, including images, video, audio, and natural language, is information that doesn't follow a predefined model or manner of organization. This data type accounts for ~80% of the world's data, and can be converted into vectors using various artificial intelligence (AI) and machine learning (ML) models.

Vector embedding

A vector embedding is a feature abstraction of unstructured data, such as a video clip, a photo, or a sound clip. Mathematically speaking, a vector embedding is an array of floating-point numbers or binaries. Modern embedding techniques, such as artificial intelligence (AI) or machine learning (ML) models are used to convert unstructured data to vector embeddings. By projecting unstructured data to a coordinate point in an n-dimensional space, approximate nearest neighbor (ANN) algorithms can be used to calculate similarities between unstructured data.

Vector similarity search

Similarity search is the process of comparing a target to a database to find objects that are most similar to it. Vector similarity search returns vectors in a database most similar to the target search vector. Approximate nearest neighbor (ANN) search algorithms are used to calculate similarity between vectors. Learn more about vector similarity search.

Tools

Milvus Insight

Milvus Insight is a graphical management system for Milvus. It features visualization of cluster states, meta management, data queries and more. Milvus Insight will eventually be open sourced.

Milvus DM

Data migration tool for Milvus 2.0 will be made available as soon as possible.

Join Our Community

Join the Milvus community on Slack to share your suggestions, advice, and questions with our engineering team.

Milvus Slack Channel

You can also check out our FAQ page to discover solutions or answers to your issues or questions.

Subscribe to Milvus mailing lists:

- Technical Steering Committee
- Technical Discussions
- Announcement

Follow Milvus on social media:

- Medium
- Twitter

---

id: roadmap.md
title: Milvus Roadmap

Milvus Roadmap

Roadmap

Milvus 2.0 time schedule

Next few big releases:

- Milvus 2.1: 2021.11
- Milvus 2.2: 2022.2

Roadmap features

DDL

Version	Feature	Owner	Status	Comment
2.0.0-rc	Supports numerical scalar data types		done	

Version	Feature	Owner	Status	Comment
2.0	Supports string data types	czs007, dragondriver	in progress	
2.0	Collection alias	lsgrep	in progress	
2.1	Supports Scalar bitmap/inverted Index for string and numeric data types		pending	
2.1	Supports data life cycle management		pending	
2.1	Automatic data partition		pending	
2.2	Collection rename		pending	

DML

Version	Feature	Owner	Status	Comment
2.0.0-rc	Supports scalar filtering		done	
2.0.0-rc	Supports for query by id		done	
2.0	Supports query by expression	fishpenguin	in progress	
2.0	Supports delete by id		pending	
2.1	Supports search by id		pending	
2.1	Vector similarity search by distance		pending	
2.2	Supports search/query result pagination		pending	
2.2	Supports upsert/primary key deduplication		pending	

Features

Version	Feature	Owner	Status	Comment
2.0.0-rc	Supports time travel to any specified point in time		done	
2.0.0-rc	Offers three levels of tunable consistency: strong, session, consistent prefix		done	
2.0	Segment compaction	sunby	pending	
2.0	Implements dynamic load balancing	sunby, xige-16	pending	
2.0	mplements dynamic handoff	xige-16, bigsheeper	pending	
2.0	Calculate distance between embeddings	yhmo	in progress	
2.1	Multi tenant support and access control		pending	
2.2	Change data capture		pending	
Long Term	Adopts incremental backup		pending	
Long Term	Supports static data encryption		pending	
Long Term	Offers embedding-as-service through data importer/transformer		pending	

Performance/Cost

Version	Feature	Owner	Status	Comment
2.0	Milvus 2.0 performance benchmark and tuning	czs007, dragondriver	pending	
2.1	Supports GPU Index building and embedding retrieval	shengjun1985	pending	
2.1	Data bulkload		pending	
2.1	Adopts cost-based query optimization algorithm to improve hybrid search efficiency		pending	
2.1	Supports ScaNN Index		pending	

Version	Feature	Owner	Status	Comment
2.2	Supports on-disk vector indexing		pending	
Long Term	Supports FPGA and other Heterogeneous hardware		pending	
Long Term	Automatic index optimization		pending	

Stability

Version	Feature	Owner	Status	Comment
2.0.0-rc	Fully managed failure recovery and service discovery		done	
2.0.0-rc	Python SDK test		done	
2.0	Chaos test	yanliang567	pending	
2.0	Pressure test	del-zhenwu	pending	
2.1	Supports segment in memory replicas		done	
2.1	Flow control && back pressure support		pending	
2.2	Query node resource isolation		pending	

Ease Of Use

Version	Feature	Owner	Status	Comment
2.0.0-rc	Helm installation		done	
2.0.0-rc	Support of Milvus Insight, a Milvus visual management tool		in progress	
2.0	Prometheus, Grafana and Jaeger support	zwd1208	in progress	
2.0	Milvus k8s operator	zwd1208, jeffoverflow	pending	
2.1	Multi datacenter deployment and multi-cloud integration		pending	
2.2	Embedded Milvus that runs on laptops		pending	
Long Term	Dynamic cluster expansion/shrink		pending	

SDK

Version	Feature	Owner	Status	Comment
2.0.0-rc	Python ORM-style APIs		done	
2.0	Merge Pymilvus ORM and Pymilvus	XuanYang-cn	in progress	
2.0	Supports NodeJs APIs	nameczz,shanghaikid	in progress	SDK is ready to use, and We will keep updating it. https://github.com/milvus-io/milvus-sdk-node
2.0	Supports Java SDK		pending	
2.0	Supports Go SDK	congqixia	in progress	
2.1	Supports Restful APIs		pending	
2.1	Supports C++ SDK		pending	
Long Term	SQL-like Query Language		pending	

Integration

Version	Feature	Owner	Status	Comment
2.0	Integrates S3		done	
2.1	Integrates Kafka		pending	
2.1	Integrates JuiceFS		pending	

Version	Feature	Owner	Status	Comment
2.1	Data stored over local/distributed filesystems		pending	
2.2	Integrates distributed KV stores such as HBase/TiKV/FoundationDB		pending	