Does Milvus support non-x86 architectures?

Milvus cannot be installed or run on non-x86 platforms.

Your CPU must support one of the following instruction sets to run Milvus: SSE4.2, AVX, AVX2, AVX512. These are all x86-dedicated SIMD instruction sets.

Where does Milvus store data?

Milvus deals with two types of data, inserted data and metadata.

Inserted data, including vector data, scalar data, and collection-specific schema, are stored in persistent storage as incremental log. Milvus supports multiple object storage backends, including MinIO, AWS S3, Google Cloud Storage (GCS), Azure Blob Storage, Alibaba Cloud OSS, and Tencent Cloud Object Storage (COS).

Metadata are generated within Milvus. Each Milvus module has its own metadata that are stored in etcd.

Why is there no vector data in etcd?

etcd stores Milvus module metadata; MinIO

are mutually independent. From the client's perspective, an insert operation is complete when the inserted data enters the message queue. However, inserted data are unsearchable until they are loaded to the query node. If the segment size does not reach the index-building threshold (512 MB by default), Milvus resorts to brute-force search and query performance may be diminished.

Can vectors with duplicate primary keys be inserted into Milvus?

Yes. Milvus does not check if vector primary keys are duplicates.

When vectors with duplicate primary keys are inserted, does Milvus treat it as an update operation?

No. Milvus does not currently support update operations and does not check if entity primary keys are duplicates. You are responsible for ensuring entity primary keys are unique, and if they aren't Milvus may contain multiple entities with duplicate primary keys.

If this occurs, which data copy will return when queried remains an unknown behavior. This limitation will be fixed in

What is the maximum amount of data that can be added per insert operation?

An insert operation must not exceed 1,024 MB in size. This is a limit imposed by gRPC.

Does collection size impact query performance when searching in a specific partition?

No. If partitions for a search are specified, Milvus searches the specified partitions only.

Does Milvus need to load the entire collection when partitions are specified for a search?

It depends on what data is needed for search. All partitions potentially show up in search result must be loaded before searching.

- For example, if you only want to search specific parition(s), you don't need to load all. Call `load_partition()` to load the intended partition(s) *then* specify partition(s) in the `search()` method call.
- If you want to search all partitions, call `load_collection()` to load the whole collection including all partitions.

Yes. If an index has been built for a collection by `create_index()` before, Milvus will automatically build an index for subsequently inserted vectors. However, Milvus does not build an index until the newly inserted vectors fill an entire segment and the newly created index file is separate from the previous one.

How are the FLAT and IVF_FLAT indexes different?

The IVF_FLAT index divides vector space into list clusters. At the default list value of 16,384, Milvus compares the distances between the target vector and the centroids of all 16,384 clusters to return probe nearest clusters. Milvus then compares the distances between the target vector and the vectors in the selected clusters to get the nearest vectors. Unlike IVF_FLAT, FLAT directly compares the distances between the target vector and every other vector.

When the total number of vectors approximately equals nlist, there is little distance between IVF_FLAT and FLAT in terms of calculation requirements and search performance. However, as the number of vectors exceeds nlist by a factor

Milvus returns success when inserted data are ingested to the message queue. However, the data are not yet flushed to the disk. Then Milvus' data node writes the data in the message queue to persistent storage as incremental logs. If `flush()` is called, the data node is forced to write all data in the message queue to persistent storage immediately.

What is normalization? Why is normalization needed?

Normalization refers to the process of converting a vector so that its norm equals 1. If inner product is used to calculate vector similarity, vectors must be normalized. After normalization, inner product equals cosine similarity.

See Wikipedia for more information.

Why do Euclidean distance (L2) and inner product (IP) return different results?

For normalized vectors, Euclidean distance (L2) is mathematically equivalent to inner product (IP). If these similarity metrics return different results, check to see if your vectors are normalized

partitions in them.

For example, let's assume you have already created 100 collections, with 2 shards and 4 partitions in 60 of them and with 1 shard and 12 partitions in the rest 40 collections. The current number of collections can be calculated as:

```
60 * 2 * 4 + 40 * 1 * 12 = 960
```

Why do I get fewer than k vectors when searching for `topk` vectors?

Among the indexes that Milvus supports, IVF_FLAT and IVF_SQ8 implement the k-means clustering method. A data space is divided into `nlist` clusters and the inserted vectors are distributed to these clusters. Milvus then selects the `nprobe` nearest clusters and compares the distances between the target vector and all vectors in the selected clusters to return the final results.

If `nlist` and `topk` are large and nprobe is small, the number of vectors in the nprobe clusters may be less than `k`. Therefore, when you search for the `topk` nearest vectors, the number of returned vectors is less than `topk`.

What is the maximum vector dimension supported in Milvus?

Milvus can manage vectors with up to 32,768 dimensions by default. You can increase the value of `Proxy.maxDimension` to allow for a larger dimension vector.

Does Milvus support Apple M1 CPU?

Current Milvus release does not support Apple M1 CPU directly. After Milvus 2.3, Milvus provides Docker images for the ARM64 architecture.

What data types does Milvus support on the primary key field?

In current release, Milvus supports both INT64 and string.

Is Milvus scalable?

Yes. You can deploy Milvus cluster with multiple nodes via Helm Chart on Kubernetes. Refer to Scale Guide for more instruction.

What are growing segment and sealed segment?

When a search request comes, Milvus searches both incremental data and historical data. Incremental data are recent

They are in the sealed segments which have been persisted in the object storage. Incremental data and historical data together constitute the whole dataset for search. This design makes any data ingested to Milvus instantly searchable. For Milvus Distributed, there are more complex factors that decide when a record just ingested can show up in search result. Learn more nuance about that at consistency levels.

Is Milvus available for concurrent search?

Yes. For queries on the same collection, Milvus concurrently searches the incremental and historical data. However, queries on different collections are conducted in series. Whereas the historical data can be an extremely huge dataset, searches on the historical data are relatively more time-consuming and essentially performed in series.

Why does the data in MinIO remain after the corresponding collection is dropped?

Data in MinIO is designed to remain for a certain period of time for the convenience of data rollback.

In Milvus, a vector similarity search retrieves vectors based on similarity calculation and vector index acceleration. Unlike a vector similarity search, a vector query retrieves vectors via scalar filtering based on a boolean expression. The boolean expression filters on scalar fields or the primary key field, and it retrieves all results that match the filters. In a query, neither similarity metrics nor vector index is involved.

Why does a float vector value have a precision of 7 decimal digits in Milvus?

Milvus supports storing vectors as Float32 arrays. A Float32 value has a precision of 7 decimal digits. Even with a Float64 value, such as 1.3476964684980388, Milvus stores it as 1.347696. Therefore, when you retrieve such a vector from Milvus, the precision of the Float64 value is lost.

How does Milvus handle vector data types and precision?

Milvus supports Binary, Float32, Float16, and BFloat16 vector types.

- Binary vectors: Store binary data as sequences of 0s and 1s, used in image

- Float16 and BFloat16 vectors: Offer reduced precision and memory usage. Float16 is suitable for applications with limited bandwidth and storage, while BFloat16 balances range and efficiency, commonly used in deep learning to reduce computational requirements without significantly impacting accuracy.

Does Milvus support specifying default values for scalar or vector fields?

Currently, Milvus 2.4.x does not support specifying default values for scalar or vector fields. This feature is planned for future releases.

Is storage space released right after data deletion in Milvus?

No, storage space will not be immediately released when you delete data in Milvus. Although deleting data marks entities as "logically deleted," the actual space might not be freed instantly. Here's why:

- **Compaction**: Milvus automatically compacts data in the background. This process merges smaller data segments into larger ones and removes logically

- **Garbage Collection**: A separate process called Garbage Collection (GC) periodically removes these "Dropped" segments, freeing up the storage space they occupied. This ensures efficient use of storage but can introduce a slight delay between deletion and space reclamation.

Can I see inserted, deleted, or upserted data immediately after the operation without waiting for a flush?

Yes, in Milvus, data visibility is not directly tied to flush operations due to its storage-compute disaggregation architecture. You can manage data readability using consistency levels.

When selecting a consistency level, consider the trade-offs between consistency and performance. For operations requiring immediate visibility, use a "Strong" consistency level. For faster writes, prioritize weaker consistency (data might not be immediately visible). For more information, refer to Consistency.

After enabling the partition key feature, what is the default value of `num_partitions` in Milvus, and why?