On December 20, 2023, StarRocks released its brand-new version 3.2. This new version makes significant enhancements to many **Killer features** of StarRocks and comprehensively upgrades the **usability** of the entire system.
Killer features such as storage-compute separation, data lake analytics, and materialized views have been further optimized. The entire operational chain, from table creation and schema change to data loading, data query, and data unloading is simplified for better usability.

A quick look

This section gives you a glimpse of some new features and major enhancements. The "Core features and enhancements" section provides more details.

- **Higher usability**
  - **Table creation**: Optimizes Random Bucketing to support automatic bucketing, supports Fast Schema Evolution and manual optimization of table structures and data distribution using Optimize Table.
  - **Data loading:** Supports continuous, large-scale data loading from AWS S3 or HDFS using PIPE, and optimizes the FILES table function.
  - **Data unloading**: Supports unloading data from StarRocks to Parquet-formatted files stored in AWS S3 or HDFS using INSERT INTO FILES, unifies the syntax for data unloading and data loading for ease of use.
  - **Data query**
    - Provides the HTTP SQL API, enabling users to access StarRocks data via HTTP and run SELECT, SHOW, EXPLAIN, or KILL operations. Users can easily access StarRocks even without a MySQL client.
    - Introduces Runtime Profile and text-based profile analysis. Users can directly analyze profiles via MySQL clients, facilitating bottleneck identification and discovery of optimization opportunities.
- **Enhanced storage-compute separation**
  - Enhances shared-data clusters (storage-compute separated) to achieve equivalent functionalities and performance as shared-nothing clusters.
  - Supports persisting indexes of Primary Key tables to local disks.
- **More powerful Data Lake Analytics**
  - Optimizes query performance in various scenarios, including file reading, predicate rewriting, partition pruning, and statistics-based query acceleration.
  - Hive Catalog allows both read and **write** operations.
  - Unified Catalog: Users can access different table formats that share a common Hive Metastore or AWS Glue. This enables more convenient and flexible data lake analytics.
- **Robust and easy-to-use asynchronous materialized views**
  - Supports partition-level incremental refresh on asynchronous materialized views created on Iceberg and Paimon catalogs, in addition to Hive catalogs.
  - Supports automatic activation of ineffective materialized views, ensuring the stability and usability of asynchronous materialized views.

    - Optimizes the data consistency of query rewrites and enables **Spill to disk** by default to reduce memory consumption, making asynchronous materialized views robust and user-friendly.

Core features and enhancements

Higher usability

In v3.2, StarRocks made significant efforts to improve system **usability**, aiming to make operations such as table creation and data loading/unloading simpler and more efficient for users. In this version, you can use the Optimize Table feature to optimize existing table structure and data distribution, effortlessly load large-scale data from cloud storage like S3 or HDFS using PIPE, and export data to S3 or HDFS using the INSERT INTO FILES command. These enhancements improve the overall usability of StarRocks throughout the entire workflow.

Table creation and schema change

- **Automatic bucketing**: In v3.1, Duplicate Key tables support the Random Bucketing feature, which saves users the effort to manually determine the number of buckets. v3.2 further optimizes the bucketing approach. The system can dynamically adjust the number of buckets based on cluster information, data volume in the loading pipe, and the loading method picked by users. This optimization significantly reduces memory usage and I/O overhead, especially in scenarios with a large number of buckets or high-frequency real-time loading. The number of buckets created is suitable for the actual data volume.
- **Support for modifying table structure and data distribution**: As query patterns evolve, users may need to modify table structures, sort keys, or partitioning schemes to suit varied performance requirements. StarRocks addresses this need by offering the Optimize Table functionality. Users can run the enhanced ALTER TABLE command to adjust table structures and reorganize data based on the latest business scenarios and performance needs. This includes adjusting bucketing methods, bucket numbers, sort keys, and the ability to adjust only bucket numbers of specific partitions.
- v3.2 introduces **Fast Schema Evolution**, offering a lightweight, efficient way for schema change operations such as adding or dropping columns. When the schema change is happening, Fast Schema Evolution synchronously modifies metadata on the frontend (FE) without the need to add new tablets. When the task is returned to the client, it signifies the completion of the ALTER operation. In most cases, adding or dropping columns can be completed **within only a few milliseconds**. Users can enable or disable Fast Schema Evolution by setting the table property `fast_schema_evolution` during table creation.
- v3.2 supports setting automatic storage cooldown time for partitioned tables. It allows automatically moving data from SSD to slower HDD hard drives. In future versions, StarRocks is also planning to support the ability to convert cold partitions to Iceberg/Hive tables, enabling more flexible hot and cold data management.

Data loading

- v3.2 supports continuous, large-scale data loading from cloud storage such as S3 or HDFS by using the **CREATE PIPE** command. When loading large-scale data, CREATE PIPE splits a large load task into multiple smaller tasks and runs them serially based on the size and number of files. This reduces task retries and minimizes resource consumption during data loading, thereby improving the stability of data loading.
    - In addition, PIPE continuously checks for new files or modifications in cloud storage directories and automatically splits the changed data files into smaller load tasks. This allows for continuous ingestion of new data into destination tables. Users do not need to maintain a separate batch task scheduling system, simplifying the maintenance of data loading.

```
Unset

CREATE PIPE user_behavior_replica
PROPERTIES (
  "AUTO_INGEST" = "TRUE"
)
AS
INSERT INTO user_behavior_replica
SELECT * FROM FILES (
  "path" =
"s3://starrocks-datasets/user_behavior_ten_million_rows.parquet",
  "format" = "parquet",
  "aws.s3.region" = "us-east-1",
  "aws.s3.access_key" = "AAAAAAAAAAAAAAAAAAAA",
  "aws.s3.secret_key" = "BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB"

        ○  );
```

- The table function FILES is further enhanced:
    - Supports loading data from Parquet or ORC files in Azure and GCP.
    - Supports extracting field information from the file path using the `columns_from_path` parameter, eliminating the need for manual configuration.
    - Supports loading complex data such as ARRAY, MAP, and STRUCT.

```
Unset
INSERT INTO user_behavior_replica
```

```
SELECT * FROM FILES (
  "path" = "s3://starrocks-datasets/*/*/*/*.parquet",
  "format" = "parquet",
  "aws.s3.region" = "us-east-1",
  "aws.s3.access_key" = "AAAAAAAAAAAAAAAAAAAA",
  "aws.s3.secret_key" =
"BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",
  "columns_from_path" = "year, month, day"
```

-);

- INSERT from FILES can deliver equivalent performance and functionalities as Broker Load but **far more superior usability**. It allows users to perform various operations on data just like they would do with a regular table. Users can easily view data using SELECT, filter data using WHERE, load data using INSERT, and directly load data using CTAS without the need to create a table.
  With the `columns_from_path` parameter, users can extract information of partitioning columns from partitioned data maintained in external systems such as Hive. Additionally, StarRocks can automatically perform Schema Merge [preview], seamlessly handling files whose table schema is changed without having users to perceive the changes.
  INSERT from FILES serves as a unified data loading mechanism and StarRocks will continue to enhance its functionality and usability.

Data unloading

Unifies the syntax of data loading (INSERT from FILES) and unloading (INSERT INTO FILES). Supports unloading data from StarRocks to Parquet-formatted files stored in AWS S3 or HDFS by using INSERT INTO FILES. After data is unloaded, you can use SELECT from FILES() to check the data.

Unset

```
INSERT INTO FILES (
  'path' = 's3://mybucket/unloading/data_folder/',
  'format' = 'parquet',
  'compression' = 'zstd',
  'aws.s3.use_instance_profile' = 'false',
  'aws.s3.access_key' = '<access_key>',
  'aws.s3.secret_key' = '<secret_key>',
  'aws.s3.region' = 'us-west-2'
```

```
)
SELECT * FROM source_table;
```

Data query

- Supports HTTP SQL API, enabling users to access StarRocks data via HTTP and execute SELECT, SHOW, EXPLAIN, or KILL operations. Users can easily access StarRocks even without a MySQL client.
- Runtime Profile and text-based profile analysis allow users to analyze profiles via MySQL clients. The analysis results help users detect the operations that consume most resources and time, and obtain detailed resource consumption information by specifying nodes in parameters. This facilitates bottleneck identification and discovery of optimization opportunities.
- The following figure shows a profile where the query is still ongoing. The small icons in the profile show clearly which operators have completed, which are still running, which are pending for running, and which stages are most time-consuming.

Storage-compute separation

The storage-compute separation architecture is continuously refined.

- v3.2 supports persisting indexes of Primary Key tables to local disks, reducing memory usage. In the future, StarRocks will support storing indexes of Primary Key tables in cloud storage. This helps eliminate fluctuations in query and loading performance caused by index rebuilding due to machine failures or replacements.
- Storage Volume has been enhanced to support parameterized configurations for HDFS-related settings. StarRocks offers rich methods to access HDFS, including simple authentication, Kerberos authentication, Namenode HA, and ViewFS. Users can configure HDFS-related information in a parameterized manner, eliminating the dependency on HDFS configuration files. With this parameterized approach, multiple Storage Volumes of HDFS types can be configured within a StarRocks cluster.
- Supports even distribution of data cache among multiple local disks.

Data lake analytics

- Many optimizations have been made to deliver extreme query performance on data lakes. These optimizations include enhancing the reading, decompression, and dictionary decoding performance of ORC/Parquet/CSV files, adaptive I/O merging, faster predicate rewriting and on-demand dictionary decoding, optimized COUNT calculation, and partition pruning for complex partition columns in the Iceberg Catalog.
- Supports collecting statistics of Hive, Iceberg, and Hudi tables and storing the statistics in StarRocks. This can feed more statistics data to the CBO to generate optimal execution plans to accelerate queries.
- **Data writing to Hive catalog**: v3.1 provides the capability to create databases/tables and write data in Iceberg Catalog. v3.2 provides the capability to create and drop databases and managed tables in Hive Catalog. Additionally, data can be written to Hive's managed tables using the INSERT or INSERT OVERWRITE statements. This means users can use StarRocks to process data in the data lake and write data from StarRocks internal tables back to the lake for other data applications to use, ensuring **Single Source of Truth** in the data lake.
- Unified Catalog: StarRocks v3.2 provides Unified Catalog, which allows users to create a universal catalog to uniformly access and manage multiple table formats (such as Hive, Iceberg, Hudi, and Delta Lake). This simplifies the creation and usage of catalogs, prevents duplicate creations for different data sources, and shields differences in table formats for users. This is particularly beneficial during data lake upgrades (for example, migrating from Hive to Iceberg or Hudi), as it allows users to enjoy a seamless and unified querying experience without perceiving the underlying migration progress.
- Supports querying system tables (Information Schema) in external catalogs. This allows users to conveniently access database and table information in external data sources, facilitating interactions between external systems (such as BI tools) and StarRocks.

Materialized view

Asynchronous materialized view

- Partition-level incremental refresh on asynchronous materialized views created on Iceberg and Paimon tables: In earlier versions, StarRocks supports partition-level refresh only for materialized views created on StarRocks **internal tables**. Since v2.5.5, StarRocks supports partition-level incremental refresh on asynchronous materialized views created on tables in Hive catalogs. v3.2 extends this capability to asynchronous materialized views created on tables in **Iceberg and Paimon** catalogs. This fine-grained refresh scheme significantly reduces resource consumption of materialized view refresh.
- Automatic activation of ineffective materialized views: In real-world scenarios, many operations may render materialized views ineffective, such as dropping and re-creating base tables, atomic swap, or schema changes in the views used for modeling. As a result, query rewrite cannot take effect. In v3.2, ineffective materialized views can be

automatically activated, making materialized views robust and query acceleration smooth and fast.
- Enhanced data consistency in transparent query rewriting: Users can tune parameter settings based on the requirements on query performance and data consistency. This is particularly useful for real-time data-based materialized views and external table-based materialized views.
- Backing up and restoring asynchronous materialized views: The lifecycle of data within the materialized view can be managed.
- More developers' features: Supports Trace Rewrite and Query Dump.
    - Trace rewrite can be used to analyze scenarios where transparent query rewriting fails, making future rewrite optimization easier.
    - The Query Dump interface is enhanced to include information about materialized views in the dump file, facilitating query analysis.

Synchronous materialized view

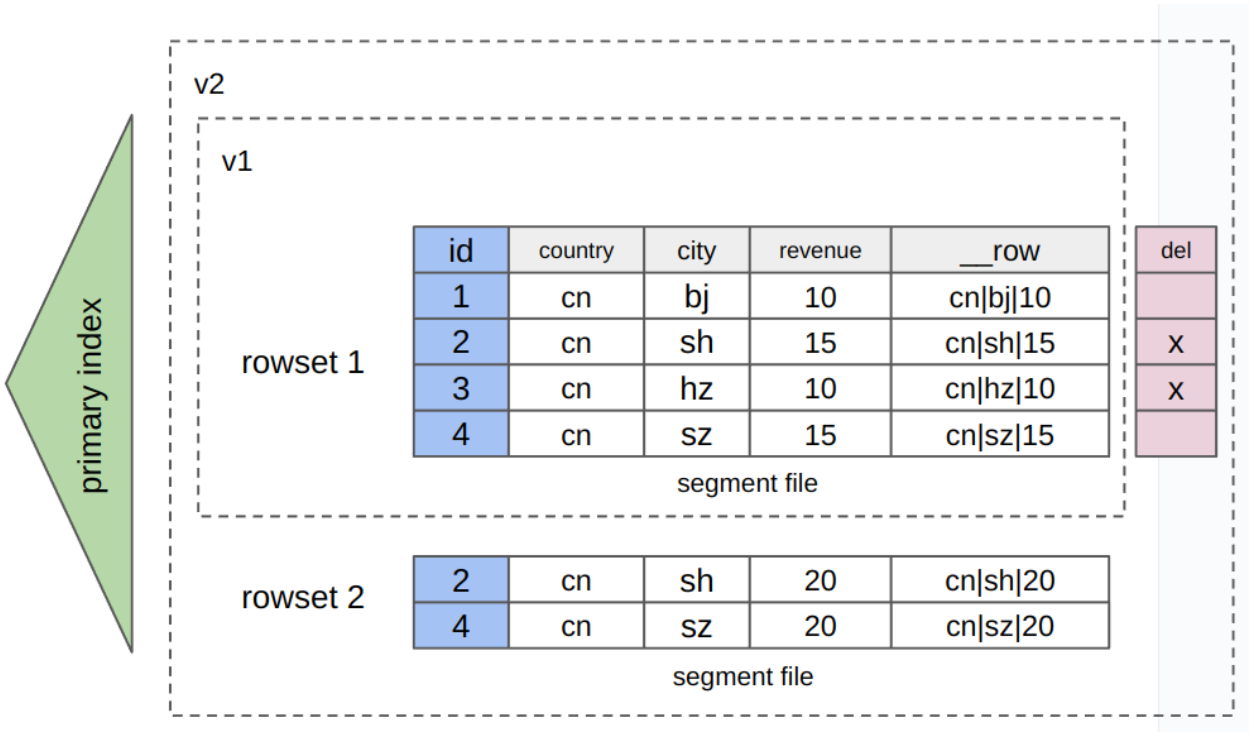
- Supports creating synchronous materialized views with the WHERE clause, in addition to the CASE-WHEN, CAST, and other mathematical expressions that were supported in v3.1.
- Supports specifying multiple aggregate columns.

Row-column mixed storage

In future 3.2 minor versions, StarRocks will introduce row-column mixed storage to Primary Key tables. Row-column mixed storage can store the entire row by adding a hidden column `__row`. Users can specify the row storage mode by configuring `"STORE_TYPE" = "column_with_row"` in PROPERTIES during table creation.
Main use cases for row-column mixed storage:

- High-concurrency point lookups based on primary keys.
- Scenarios where partial updates are frequently performed.

Row-column mixed storage enables StarRocks to deliver more efficient concurrent query capabilities and data update capabilities, while retaining the efficient analytical capabilities of the original columnar storage.



Other enhancements

- Supports Prepared Statement, which delivers better performance for processing high-concurrency point queries. It can also effectively prevent SQL injection.
- Optimized the persistent index for Primary Key tables by improving memory usage logic while reducing I/O read and write amplification.
- Supports data re-distribution across local disks for Primary Key tables.
- Supports many SQL functions, including string functions, date and time functions, and window functions.
- Optimized StarRocks' compatibility with Metabase and Superset. Supports integrating them with external catalogs.