StarRocks Technical Overview

A Linux Foundation Project

Albert Wong, albert.wong@celerdata.com

StarRocks

Agenda

- State of the OLAP software landscape
- What is StarRocks
- StarRocks' Release Timeline
- Major Features in StarRocks

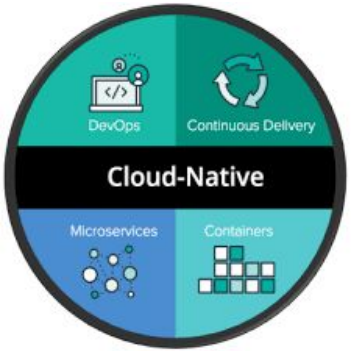
State of the OLAP software landscape

Trends in OLAP databases

Online analytical processing (OLAP) databases are evolving rapidly to meet the demands of modern data analytics. Here are some of the key trends in OLAP databases:

1 **Cloud Native**

* Separation of Compute and Storage
* Containers
* k8s operator



2A **Sub-second vs. Second/Minute Query Response Time**

2B **Streaming vs. Batch Data**

2C **Mutable vs. Immutable Data**

2D **Remote (Object) Storage vs. Local (SSD) Storage**

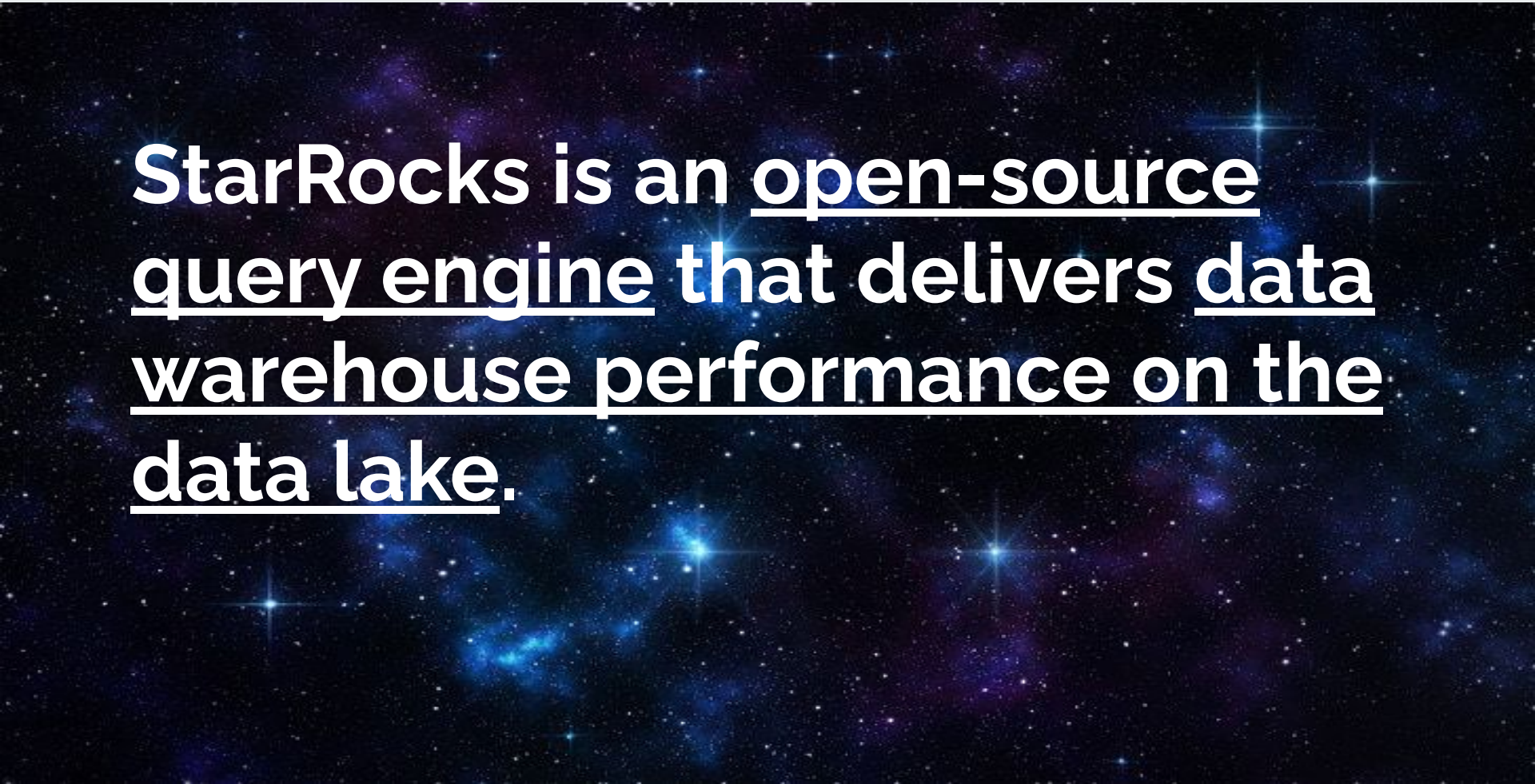
2E **Open Table Format vs. Product Native Storage Format**

3 **Data Warehouse vs. Data Lake vs. Data Lakehouse**

Trends in OLAP databases

Open Lakehouse vs Proprietary / Hybrid Lakehouse.

	Proprietary / Hybrid	Open Storage	Open
Compute	snowflake	databricks *photon*	StarRocks trino
Data Catalog	snowflake	databricks HIVE	HIVE
Table Format	snowflake ICEBERG	DELTA LAKE	HIVE ICEBERG Apache hudi DELTA LAKE Apache Paimon
Storage Format	snowflake Parquet	Parquet	Parquet

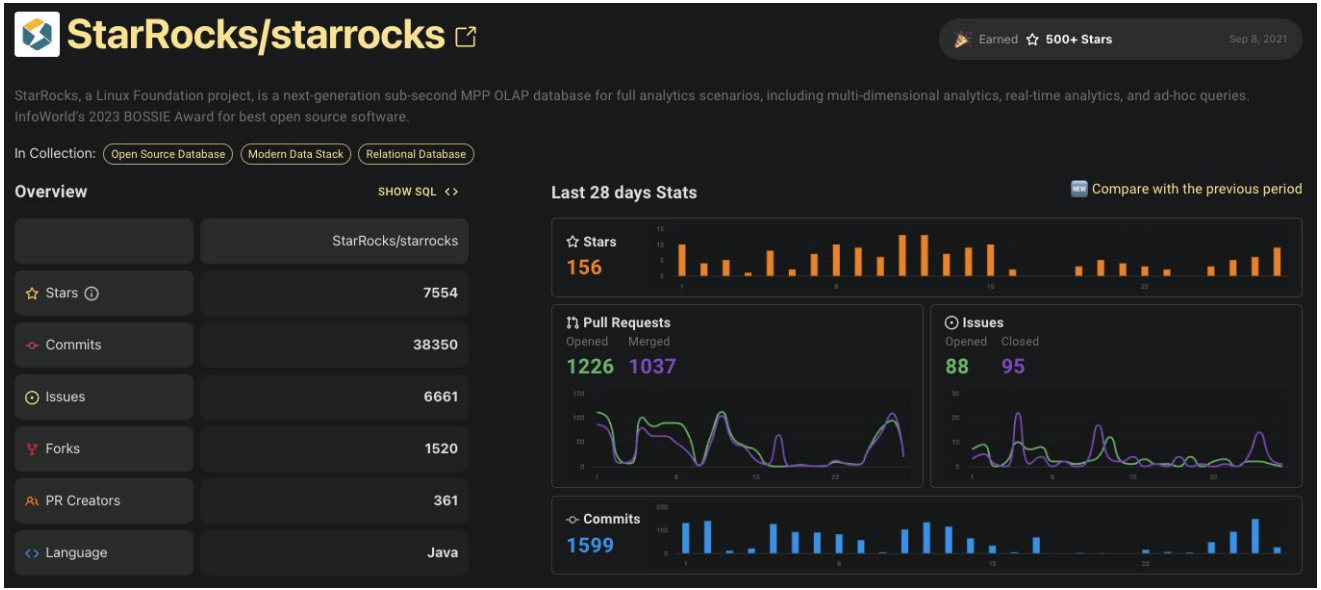

StarRocks is an open-source query engine that delivers data warehouse performance on the data lake.

StarRocks Community

LeetCode

Fanatics

Trip.com



SHEIN

Shopee

Tencent

DiDi

JD.COM

7500+ Github Stars

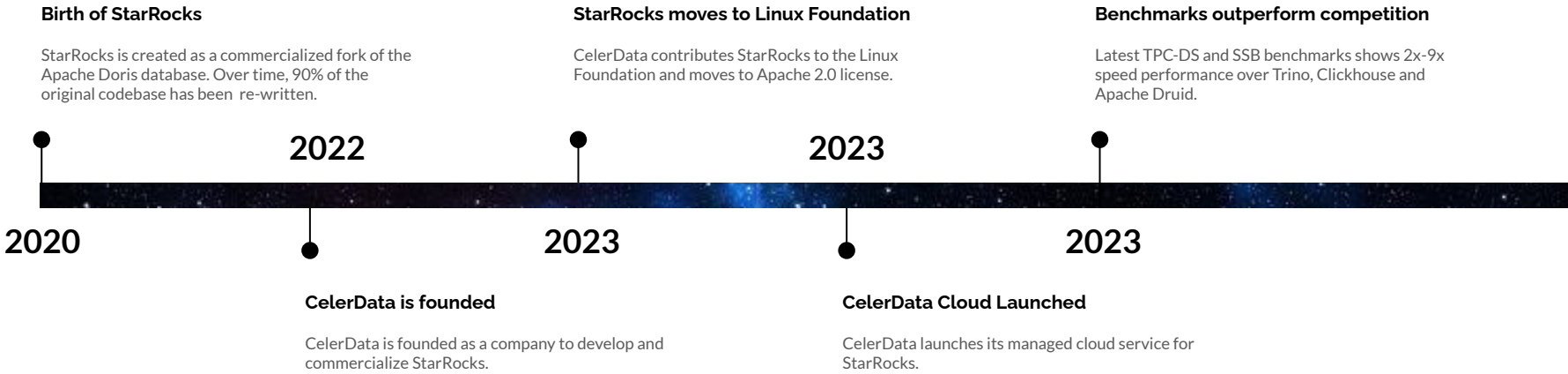
350+ Contributors

18,000+ Community Members

As of Feb 2024

History of StarRocks and CelerData

StarRocks was designed to address the challenges of real-time analytics, including the need to support high concurrency, low latency, and a wide range of analytical workloads. StarRocks also offers a number of features that are not available in other real-time analytics databases, such as the ability to query data directly from data lakes.



Birth of StarRocks

StarRocks is created as a commercialized fork of the Apache Doris database. Over time, 90% of the original codebase has been re-written.

CelerData is founded

CelerData is founded as a company to develop and commercialize StarRocks.

StarRocks moves to Linux Foundation

CelerData contributes StarRocks to the Linux Foundation and moves to Apache 2.0 license.

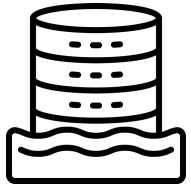
CelerData Cloud Launched

CelerData launches its managed cloud service for StarRocks.

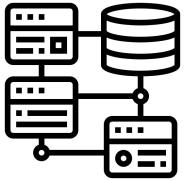
Benchmarks outperform competition

Latest TPC-DS and SSB benchmarks shows 2x-9x speed performance over Trino, Clickhouse and Apache Druid.

StarRocks is an open-source query engine that delivers data warehouse performance on the data lake.

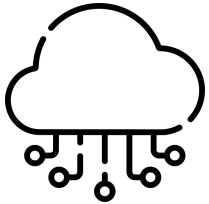


Directly query data on data lake

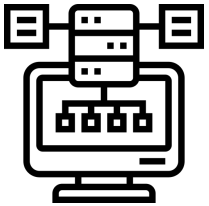


Sub-second joins and aggregations on billions of rows

Hundreds of thousands of concurrent end-user requests



Cloud Native w/ separating Compute and Storage tiers



JOIN performant at scale; denormalization optional

mysql protocol with Trino dialect

StarRocks Use Cases: User-facing analytics

User-facing analytics (UFA) is a rapidly growing field that is transforming the way businesses deliver insights to their users. UFA empowers users to explore and analyze data for themselves, without the need for technical expertise. This can lead to a number of benefits, such as:

1. Improved decision making
2. Increased user engagement
3. Reduced reliance on IT

Key trends in user-facing analytics:

Self-service Analytics

Embedded Analytics

Real-Time Analytics

Augmented Analytics

Conversational Analytics

StarRocks Use Cases: Real-time analytics

Real-time analytics is the process of collecting, processing, and analyzing data as it is generated, in order to gain insights into the present state of a system or process. This can lead to a number of benefits, such as:

1. Make better decisions
2. Increased user engagement
3. Staying ahead of the competition

Key trends in real-time analytics:

Rise of streaming data

Growth of Edge Computing

Increasing use of machine learning

Democratization of real-time analytics

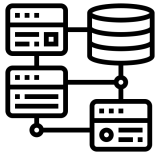
StarRocks Use Cases: Data Lakehouse

A data lakehouse is a revolutionary data architecture that merges the best of both data lakes and data warehouses.

1. Democratized Data Access
2. Increased agility and insights
3. Reduced costs and complexity
3. Faster and more accurate analytics

Key trends in data lakehouse:

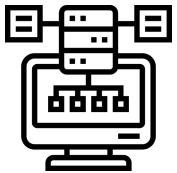
Directly query data on data lake



Sub-second joins and aggregations on billions of rows

Hundreds of thousands of concurrent end-user requests

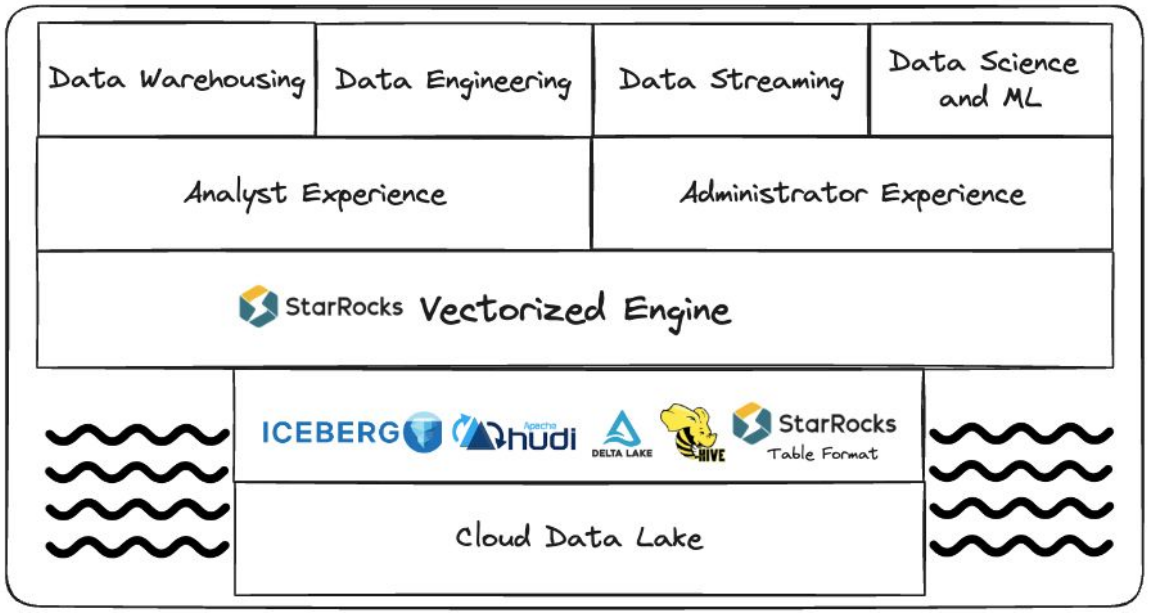
Cloud Native w/ separating Compute and Storage tiers



JOIN performant at scale; denormalization optional

StarRocks Architecture Overview

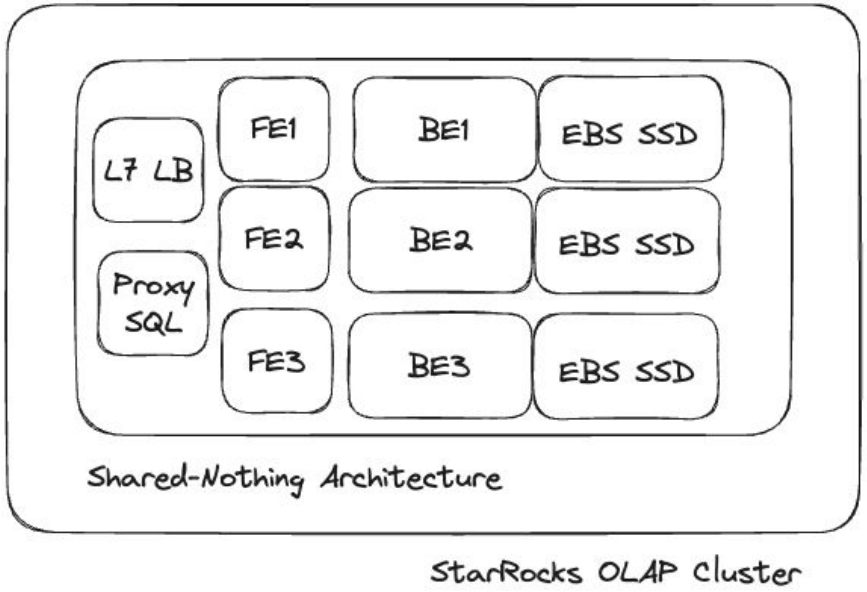
Linux Foundation project with Apache 2.0 license.

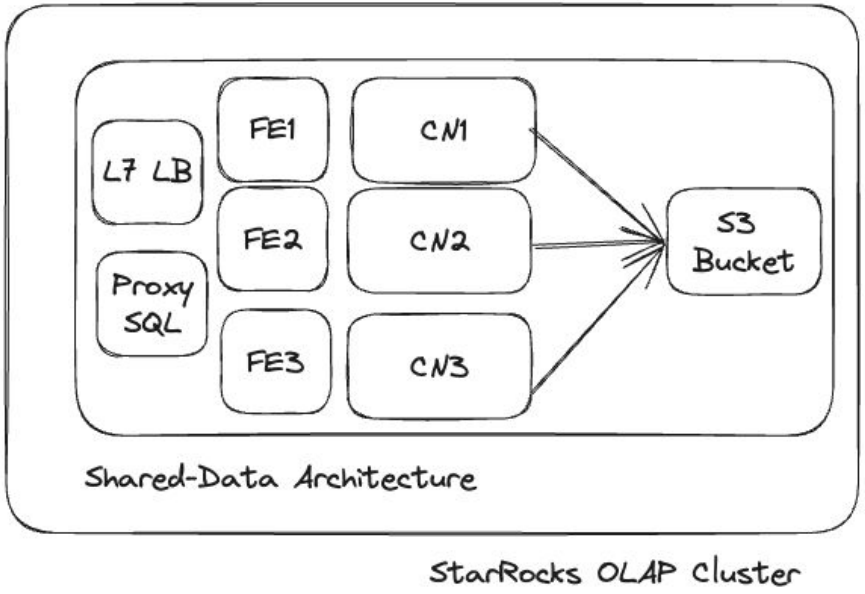


- Seamless integration with the Ecosystem
- Ease of Use
- Real-world Performance
- Open Source OLAP compute engine
- Open Table Formats as the Foundation
- Support for Open Storage
- Separated compute and storage architecture
- Cloud Native with k8s Operator

More diagrams: https://github.com/StarRocks/starrocks-reference-architecture

Two Deployment Architectural Choices

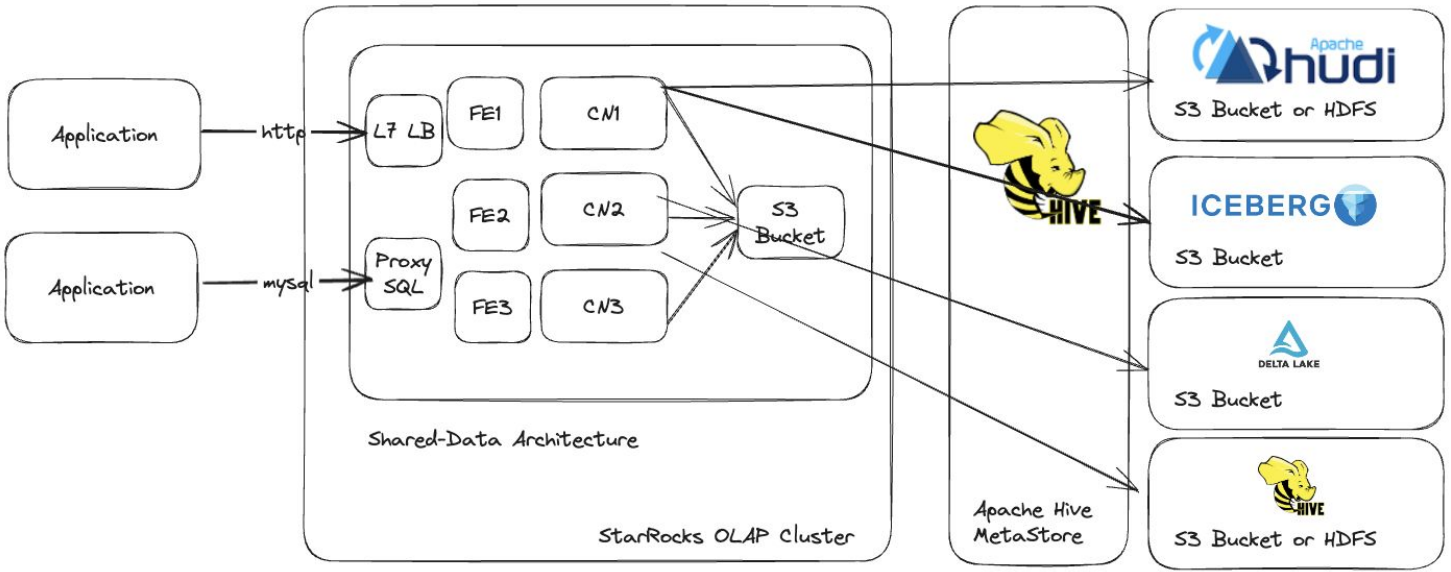




More diagrams: https://github.com/StarRocks/starrocks-reference-architecture

StarRocks with Open Data Lake

StarRocks can access multiple open table formats at the same time and even be able to create a materialized view across all of them.



More diagrams: https://github.com/StarRocks/starrocks-reference-architecture


StarRocks past and future

StarRocks Technical Features

1.x (2020-2021)

OLAP for Real Time Analytics

- Vectorized Execution Engine
- Cost Based Optimizer
- Vectorized ingestion
- Apache Hive support
- Bitmap optimization
- TopN optimization
- Lateral JOIN
- Fast Decimal support
- Tableau compatibility
- Global runtime filter
- Primary Key Table

2.x (2022-2023)

OLAP for Data Lake

- Global low-cardinality dictionary
- Pipeline Engine
- Apache Iceberg Support
- Resource Group
- Java UDF
- JSON data type support
- Partial update feature
- JDBC external catalog support
- Primary key Index
- Fully support delete and update operations
- Multi-table materialized view
- More table statistics including histogram
- Compute node on k8s
- Separation of storage and compute
- Local cache for open table formats on data lake
- Semantic cache
- Fully support RBAC
- Map/Struct data type
- Lambda function

3.0 (2023)

Shared Data Arch.

- Shared Data Architecture
- New RBAC privilege system
- Spill to disk
- Fully support for update
- Support more complete UPDATE and DELETE syntax in primary key tables
- Presto/Trino compatible dialect
- Broadcast JOIN and Bucket Shuffle JOIN can use query cache
- Global UDFs

3.1 (2023)

Shared Data Arch. Optimization

- Primary Key table support in Shared Data
- Auto_Increment column attribute
- Automatic partition creation during load
- Support Apache Iceberg v2 tables
- Random bucketing
- FILES keyboard
- Generated columns
- Support loading data into MAP and STRUCT data types
- Support nesting Fast Decimal values in ARRAY, MAP and STRUCT
- Optimized creation of async materialized view
- Optimized query rewrite with async materialized views
- Optimized refreshing of aysn materialized views
- Optimized caching, and query logic for StarRocks table format and Apache Iceberg

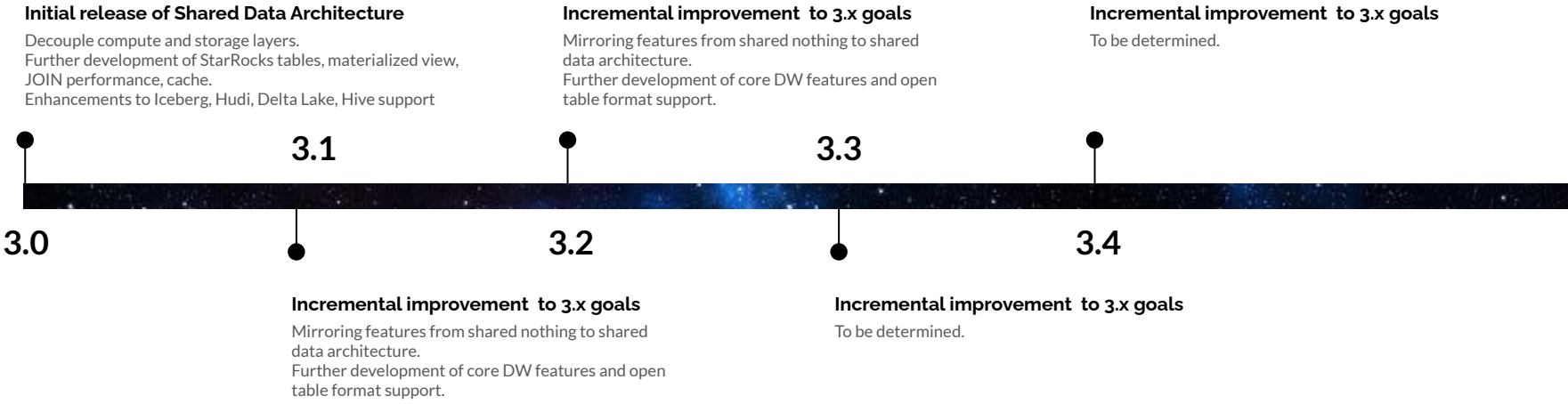
3.2 (2023)

Shared Data Arch. Optimization v2

- Persisting Primary Key table indexes to local disk
- Spill to Disk enabled by default for async materialized views
- Support creating, dropping database and managed tables in Apache Hive catalogs
- Unified Catalog
- Supports Information Schema for external tables
- Enhanced Files()
- Support unloading data from StarRocks to parquet
- Supports manual optimization of table structure and data distribution strategy
- Continuous data loading using PIPE
- Support HTTP SQL API
- Runtime Profile and text-based profile analysis commands
- Support access control through Apache Ranger
- Optimized open file format readers
- Added data consistency features for async materialized view
- Hot and warm storage support
- Fast Schema Evolution
- Dynamically adjusting number of tables
- Data redistribution across local disks for primary key tables

StarRocks 3.x series roadmap

The goal of the 3.x series roadmap is to 1) Build more and optimize core data warehouse features, 2) have feature parity between the the shared-nothing architecture and shared-data architecture and 3) be able to query the StarRocks table format and all the popular open table formats such as Apache Iceberg, Apache Hudi, Apache Hive, Delta Lake and Apache Paimon.

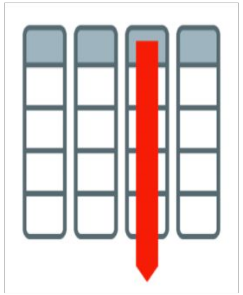



Major Features in StarRocks

Vectorized Query Engine with SIMD

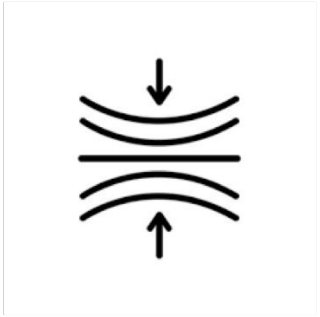
Modern CPUs have vectorized instruction sets, which can perform operations on multiple data elements simultaneously which means faster queries by 3x to 5x over non-SIMD databases.

Scan



- Reducing IO
- SIMD filter

Compression



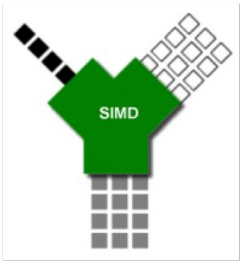
- 4-8x compression rate
- Adaptive codec selection

Indexes

- Min/max
- Ordinal index
- Secondary index

IO acceleration

Vectorized



- SIMD Instructions
- CPU prefetch and cache friendly
- Branch prediction friendly

CPU acceleration

Table Type Support

Tables are units of data storage. Understanding the table structure in StarRocks and how to design an efficient table structure helps optimize data organization and enhance query efficiency.

Table Type	Duplicate Key	Append Only	✅
	Aggregate Key	Append Only	✅
	Primary Key	All CRUD	✅

Types of Tables supported

- Duplicate Key
  - Analyze raw data, such as raw logs and raw operation records.
  - Query data by using a variety of methods without being limited by the pre-aggregation method.
  - Load log data or time-series data. New data is written in append-only mode, and existing data is not updated.
- Aggregate Key
  - Help website or app providers analyze the amount of traffic and time that their users spend on a specific website or app and the total number of visits to the website or app.
  - Help advertising agencies analyze the total clicks, total views, and consumption statistics of an advertisement that they provide for their customers.
  - Help e-commerce companies analyze their annual trading data to identify the geographic bestsellers within individual quarters or months.
- Primary Key
  - Stream data in real time from transaction processing systems into StarRocks.
  - Join multiple streams by performing update operations on individual columns.

JOIN performance at scale

Simply your data engineering pipeline and infrastructure by using JOINS; denormalization is optional.

Types of JOINS supported

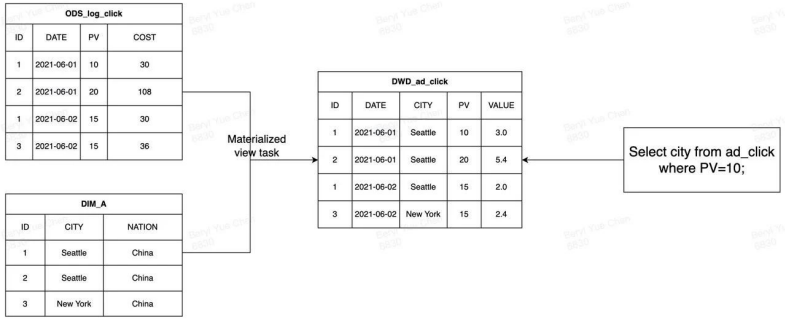
- CBO will do intelligent Join reorder and Join method selection
- Starrocks can join 100 million rows of data per second using only 1 CPU. Details at https://www.starrocks.io/blog/benchmark-test

SQL JOINS	Inner Join	✅
	Left Join	✅
	Right Join	✅
	Full Join	✅
	Cross Join	✅
	Semi Join	✅
	Anti Join	✅
SQL JOINS Optimization Technique	Broadcast Join	✅
	Shuffle Join	✅
	Bucket Shuffle Join	✅
	Co-Located Join	✅
	Replicated Join	✅
	Local Join	✅

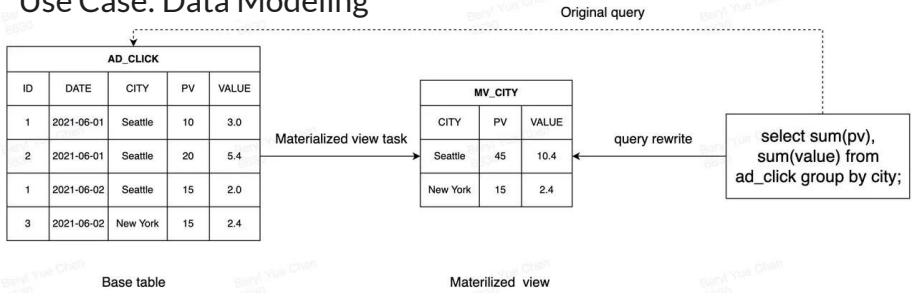
Materialized View

Materialized views can significantly improve query performance by pre-computing common aggregations.

Use Case: Query Acceleration



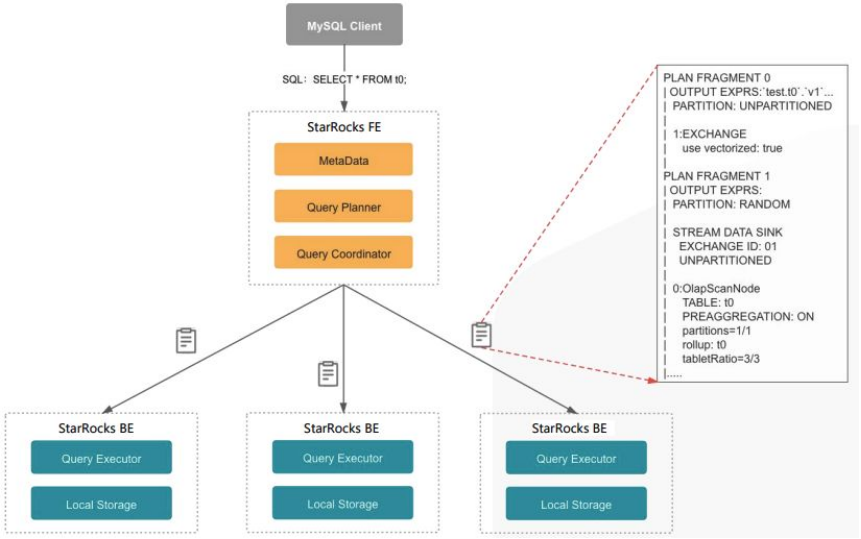
Use Case: Data Modeling

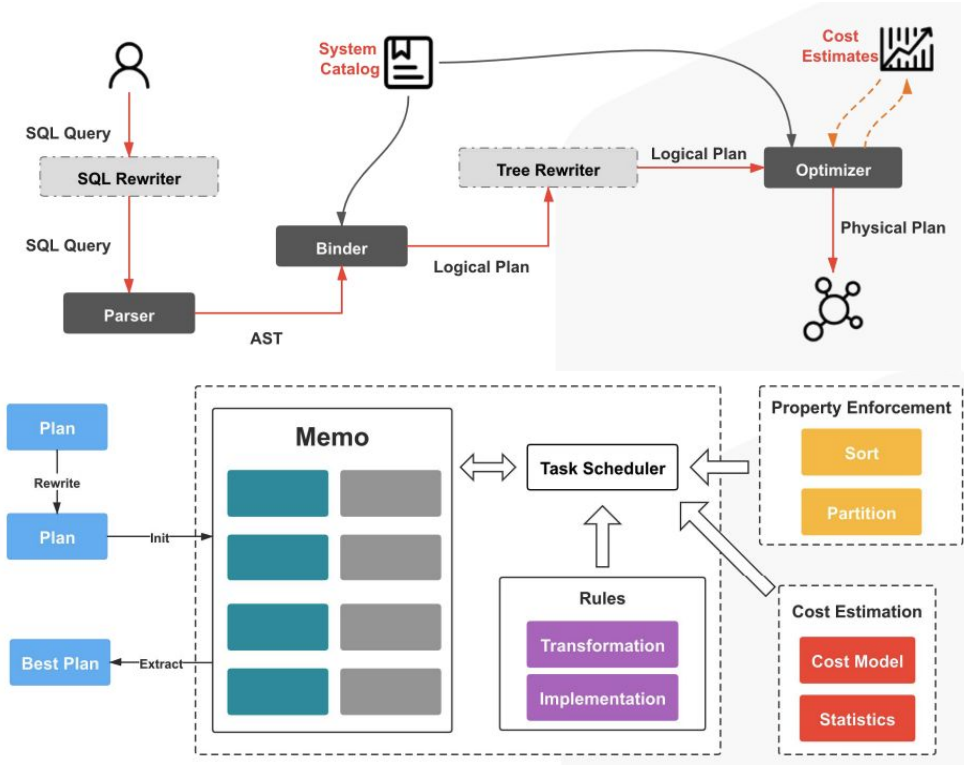


Transparent Speedup (Core Functionality)	PROJECT	✅
	AGGREGATE	✅
	JOIN	✅
	Outer-Join	✅
	View-Delta-Join	✅
	PARTIAL-UNION	✅
	NESTED MV	✅
	View-Based	✅
Incremental Refresh (Core Functionality)	Auto Refresh	✅
	Scheduled Refresh	✅
	Partition-Wise	✅

SQL Hybrid-Based Optimizer

Analyzes a SQL query and chooses the most efficient execution plan by estimating the cost of different potential plans



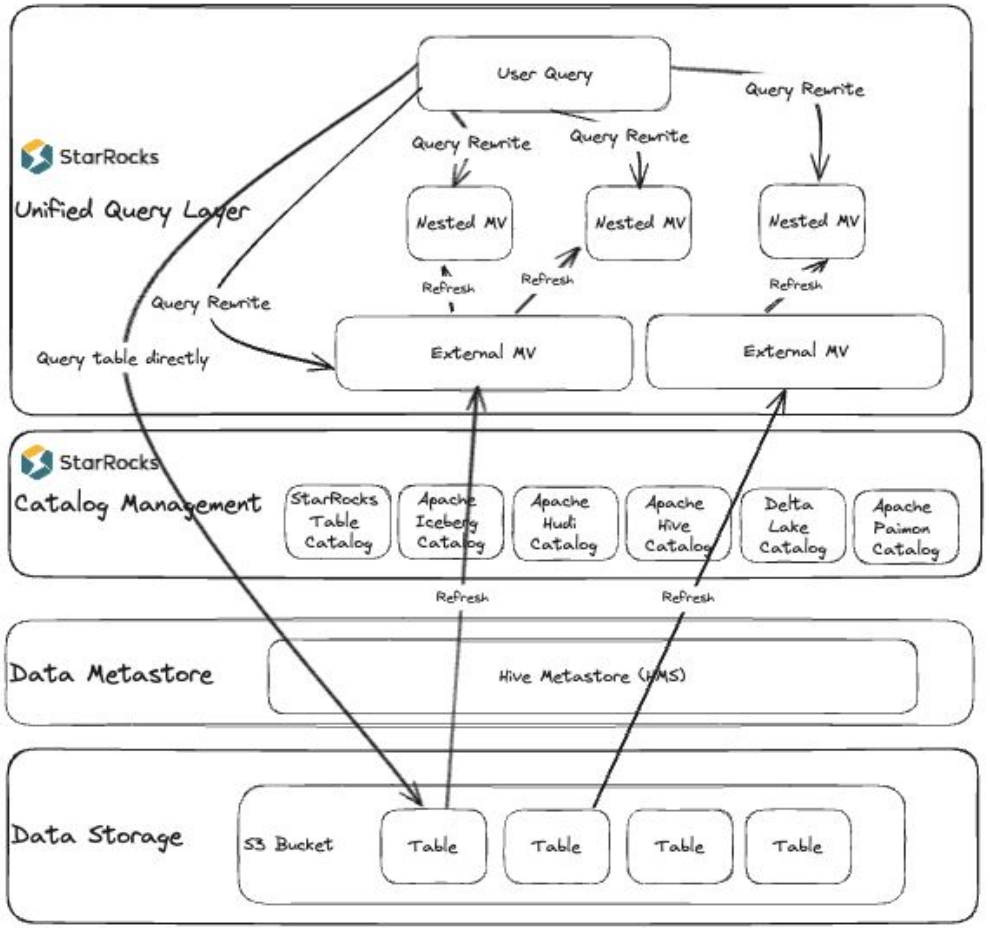


Query Rewrite

Technique used to optimize database queries without the user needing to change their original query.

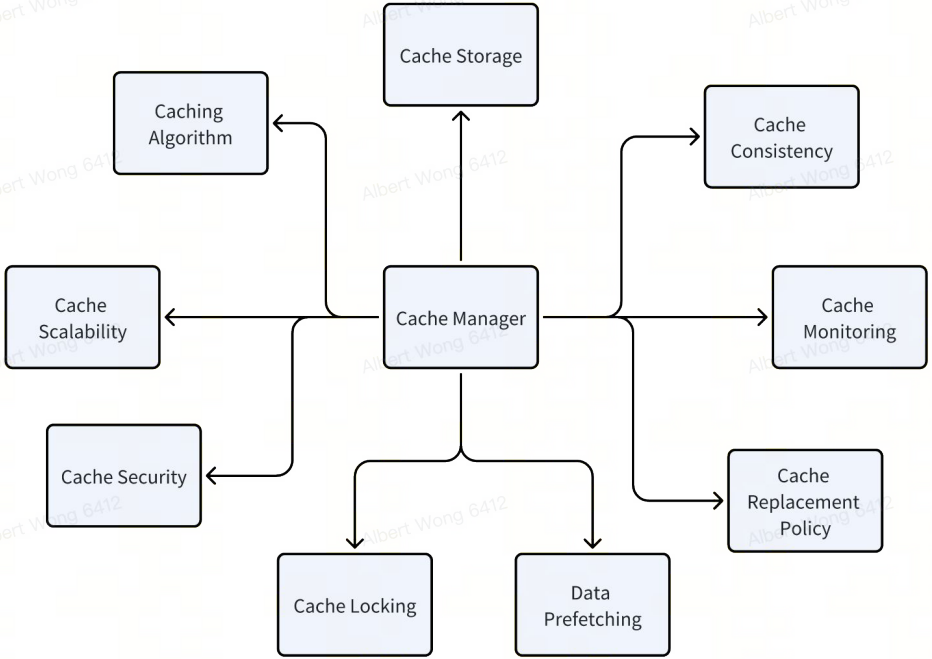
Use Case: Semantic Layer

- Targeted at Select - Projection - Join - Aggregation (SPJA) query pattern
- Up to 10x performance increase



Cache System

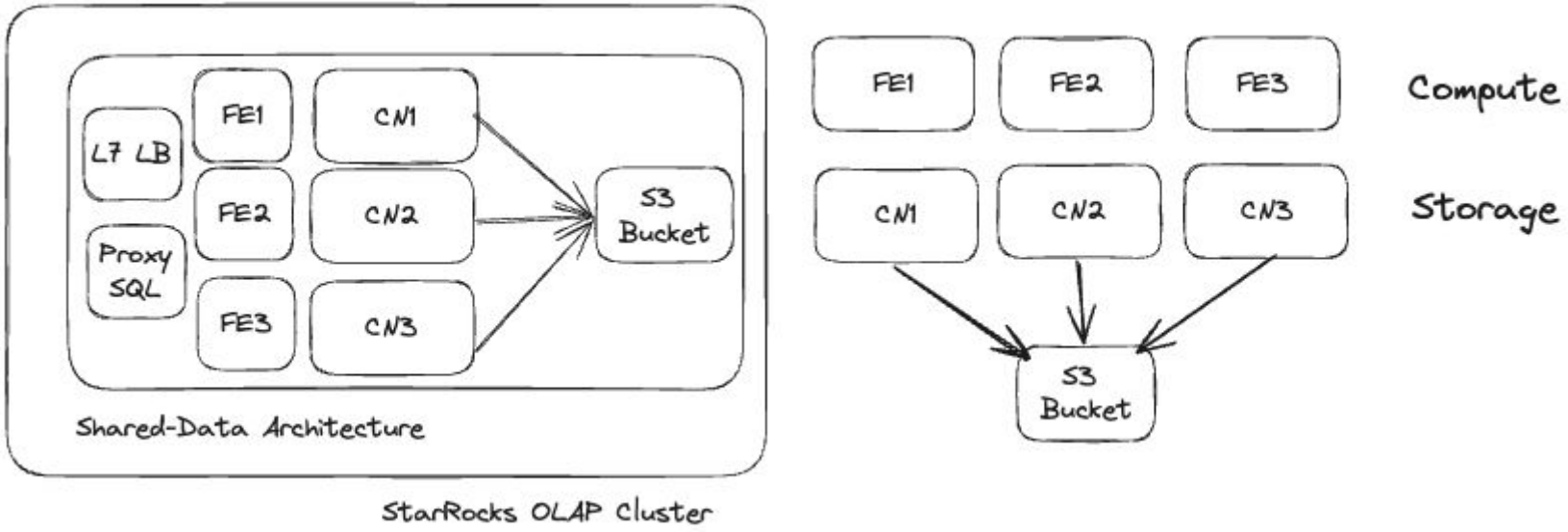
Cache allows you to pull the data from memory instead of storage which can improve query efficiency by 3x to 17x.



Transparent Speedup (Cache Functionality)	Metadata	✅
	Query	✅
	Page	✅
	Data	✅

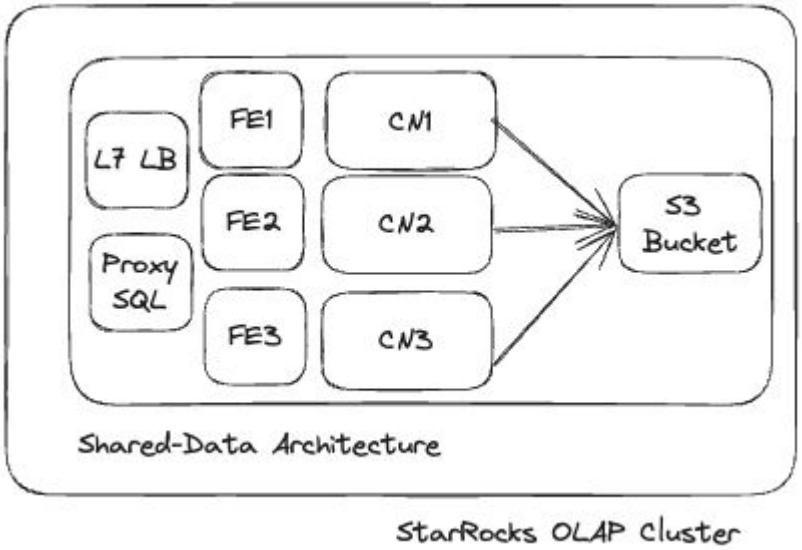
Separated compute and storage architecture

Design approach for databases and data platforms that decouples the processing power (compute) from the data storage layer.



High Availability

Redundant components and data allows the database to respond even when there is failure.



Service Availability	FE	✅	Additional Nodes
	CN	✅	Additional Nodes
	MySQL	✅	3rd party ProxySQL
	HTTP Services	✅	3rd party Load Balancer
	S3 Bucket	✅	3rd party vendor
	Data Files	✅	3rd party vendor S3 Bucket Vendor

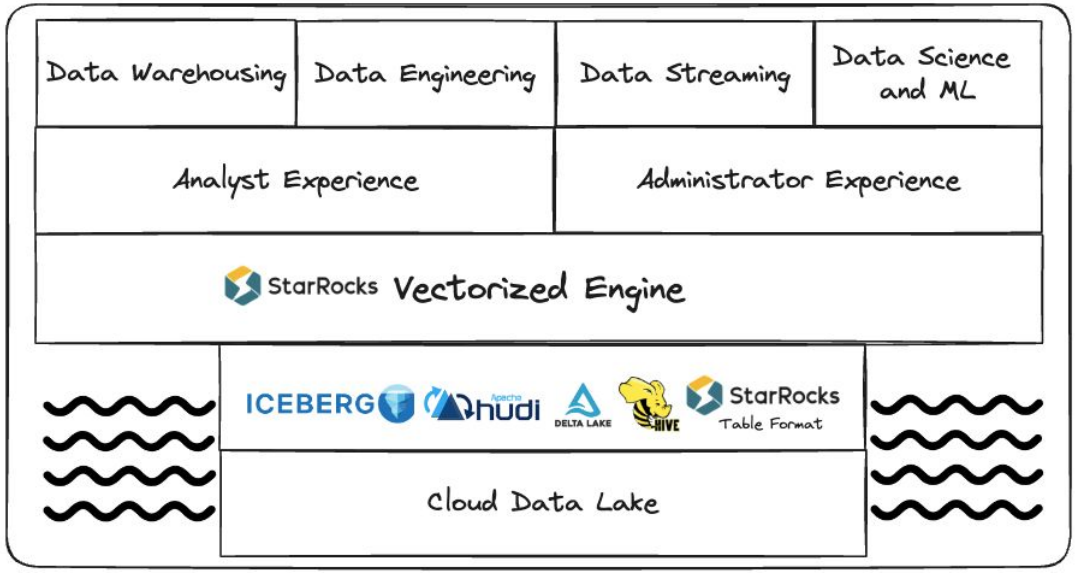
Columnar Storage

Stores data in a table by separating each column into its own continuous block instead of grouping entire rows together.

Columnar Storage Formats	StarRocks Table Format
	Apache Iceberg
	Apache Hudi
	Apache Hive
	Delta Lake
	Apache Paimon

Support for Open Table Formats

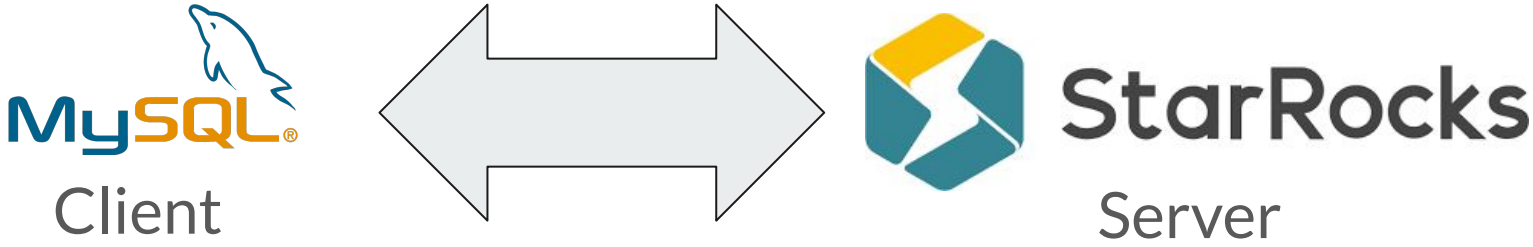
Open Table Formats allow users to extract more value from their data while maintaining flexibility and control.



Open Table Formats	StarRocks Table Format	✅ (Read/Write)
	Apache Iceberg	✅ (Read/Write)
	Apache Hudi	✅ (Read)
	Apache Hive	✅ (Read/Write)
	Delta Lake	✅ (Read)
	Apache Paimon	✅ (Read)

SQL Connectivity through MySQL wire protocol support with Trino dialect

Communicate with StarRocks through MySQL statements and utilities. Also understands the Trino SQL dialect.

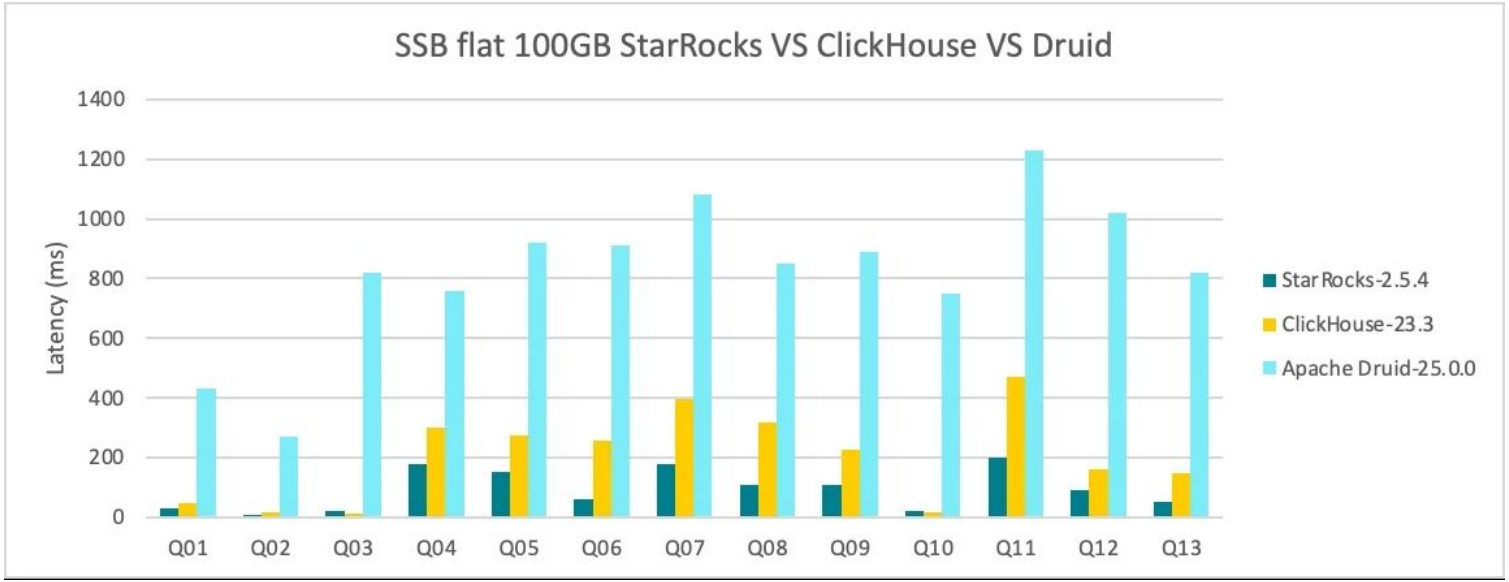


Apache Flink kafka Superset Airbyte dbt

Benchmarks and Community References

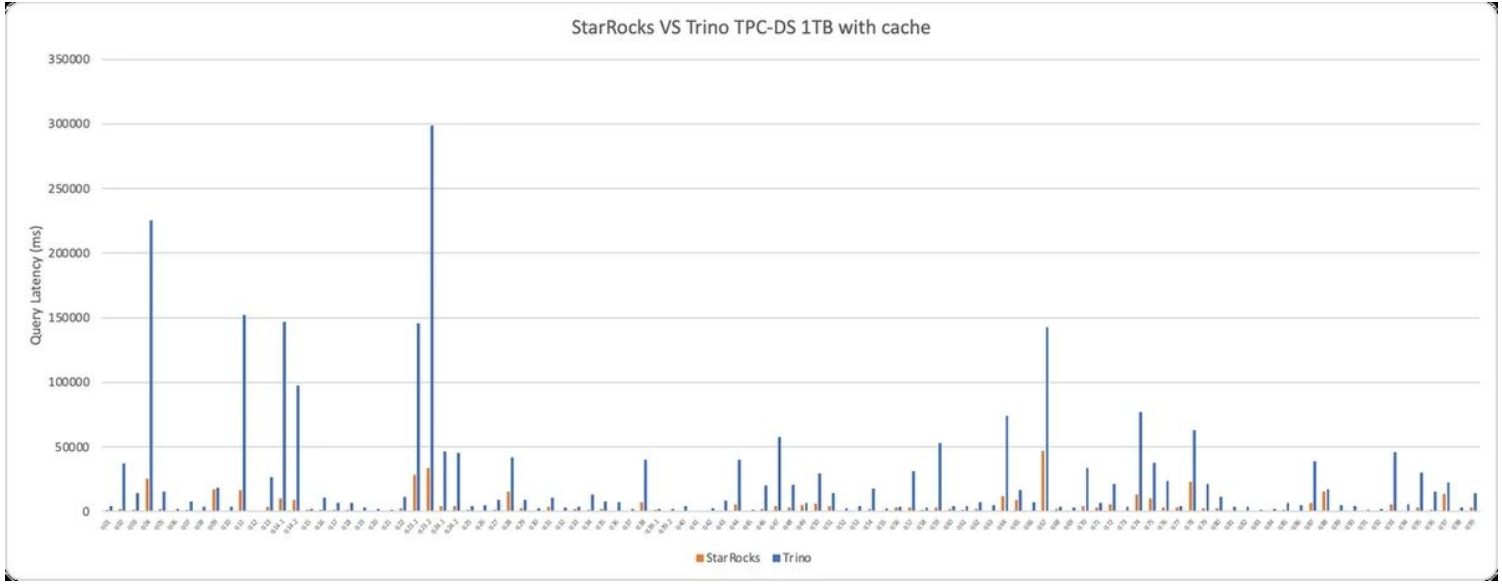
Benchmark

StarRocks Offers 2.2x Performance over ClickHouse and 8.9x Performance over Apache Druid® in Wide-table Scenarios Out of the Box using product native table format.



Benchmark

StarRocks Delivers 5.54x Query Performance over Trino in Multi-table Scenarios using Apache Iceberg table format with Parquet files.



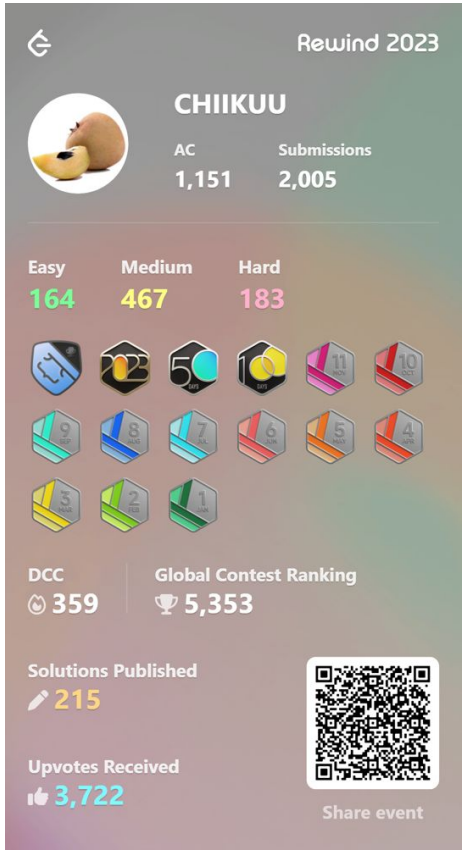
Use Case: User Analytics at LeetCode

LeetCode

LeetCode's current data warehouse, built on an OLTP database, was struggling under the weight of terabytes of user activity data. Using this OLTP database, queries took ages, impacting user experience and hindering LeetCode's ability to analyze trends and optimize the platform. Scaling up the existing system proved costly and unsustainable.

StarRocks Solution:

- Queries 100x Faster: Complex analytics that previously took hours now finished in seconds, empowering LeetCode to gain real-time insights into user behavior and platform performance. Additionally, some queries that couldn't run in the OLTP system were able to run successfully in StarRocks.
- Unlimited Scalability: StarRocks' horizontal scaling effortlessly accommodated LeetCode's growing data volume, eliminating concerns about future bottlenecks.
- Cost Savings of 80%: Compared to the a similar managed OLAP solution on GCP, StarRocks delivered significant cost savings, allowing LeetCode to reinvest in platform development and user experience.

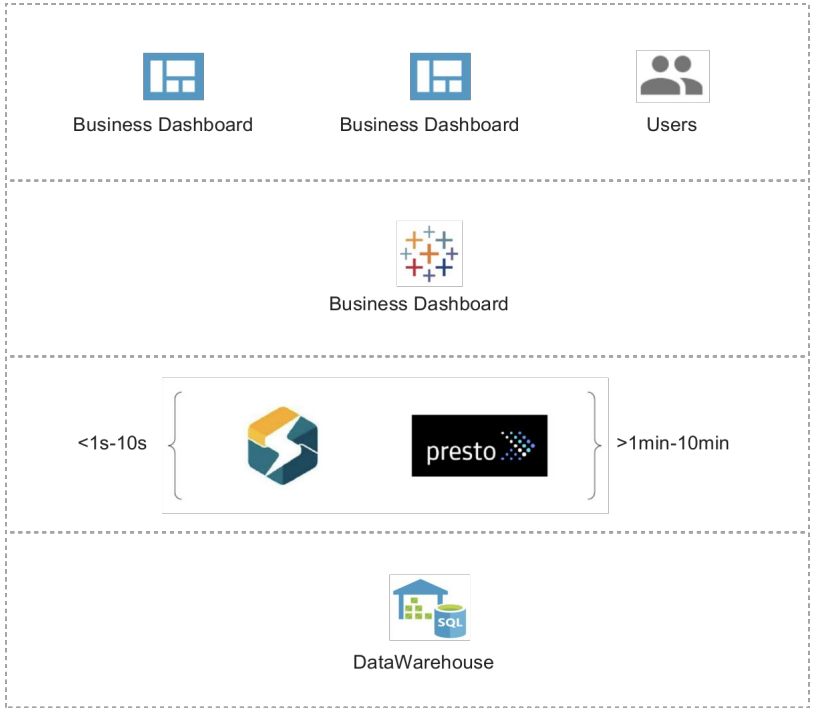


Use Case: Tableau Dashboard at Airbnb

The Airbnb Tableau Dashboard project is designed to serve both internal and external users by providing interactive dashboards. It requires a quick response to user queries. However, the query latency of previous solutions is over 10 mins, which is not acceptable. This project was just suspended until StarRocks is adopted.

StarRocks Solution:

- StarRocks can directly connect and works very well with Tableau.
- 3 tables (0.5B rows, 6B rows, 100M rows) + 4 joins + 3 distinct count + JSON functions and regex at same time, response time just 3.6s.
- Reduce the query response time from mins level to sub-seconds level.

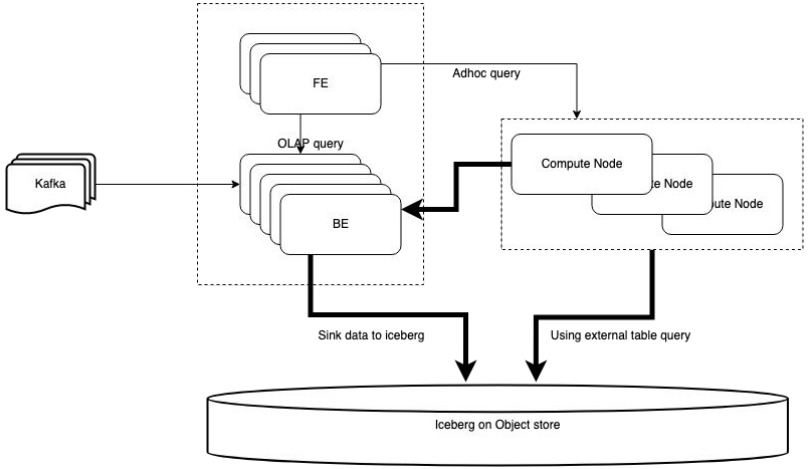


Use Case: Game and User Behavior Analytics at Tencent IEG

- 400+ game data analysis and user behavior analysis
- Operation reports need to be real-time.
- Using ClickHouse for real-time analysis and Trino for Ad-hoc before, but they want to integrate them all.
- Using Iceberg + COS store, need better performance.
- Need elastic in ad-hoc query to deduce cost.

StarRocks Solution:

- Using StarRocks Primary key to solve update problem.
- Using compute node on k8s to auto-scaling.
- Get much more performance in ad-hoc query.

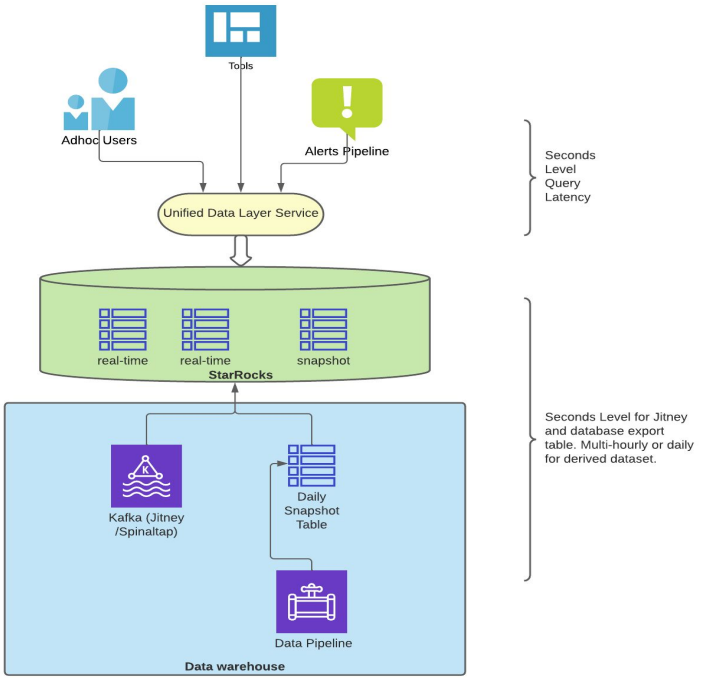


Use Case: Trust Analytics at Airbnb

To enhance security, Airbnb needs a real-time fraud detection system (Trust Analytics) to identify various attacks and take actions ASAP. This system must support Ad-Hoc query and real-time update.

StarRocks Solution:

- StarRocks hosts real-time updated datasets via Primary Key.
- Dataset import from Kafka has a sub-minute delay.
- StarRocks provides second-level query latency for complex joins.
- Alerting can be achieved by just running a SQL query regularly.



Thank you.

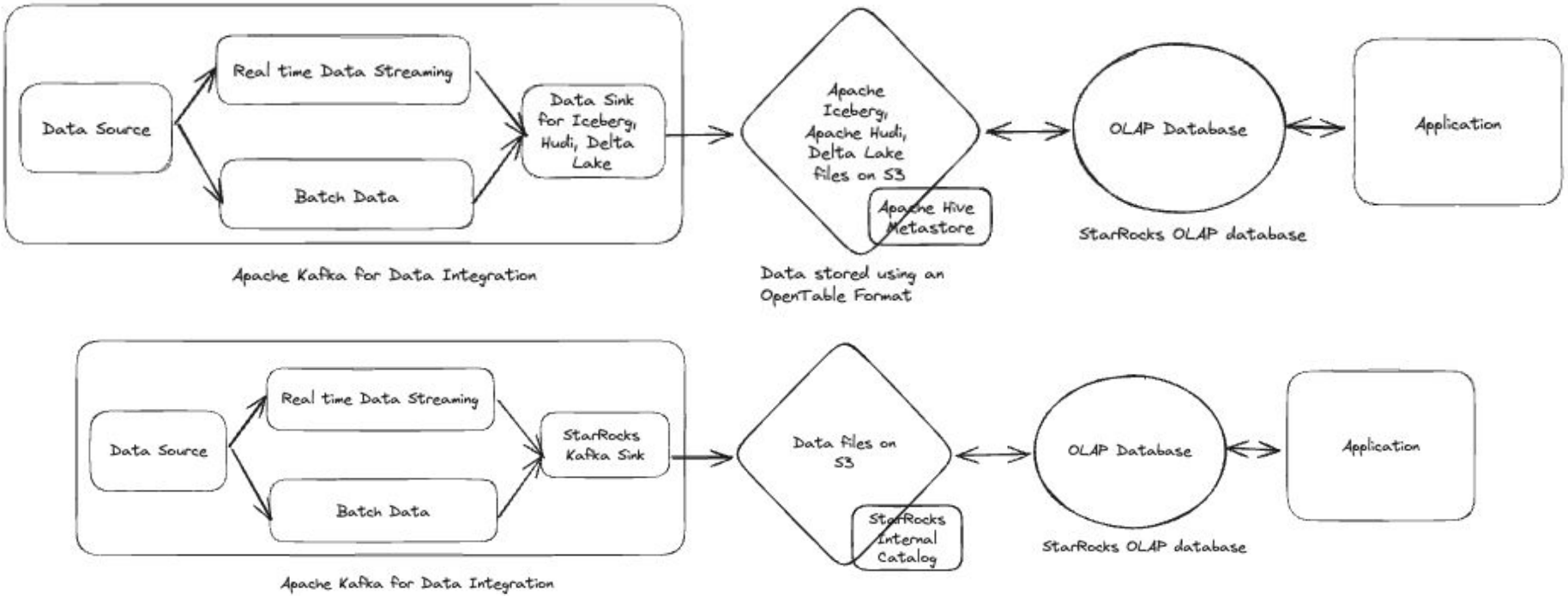
• Community	starrocks.io
• Enterprise	celerdata.com
• Managed Service	cloud.celerdata.com

Credits

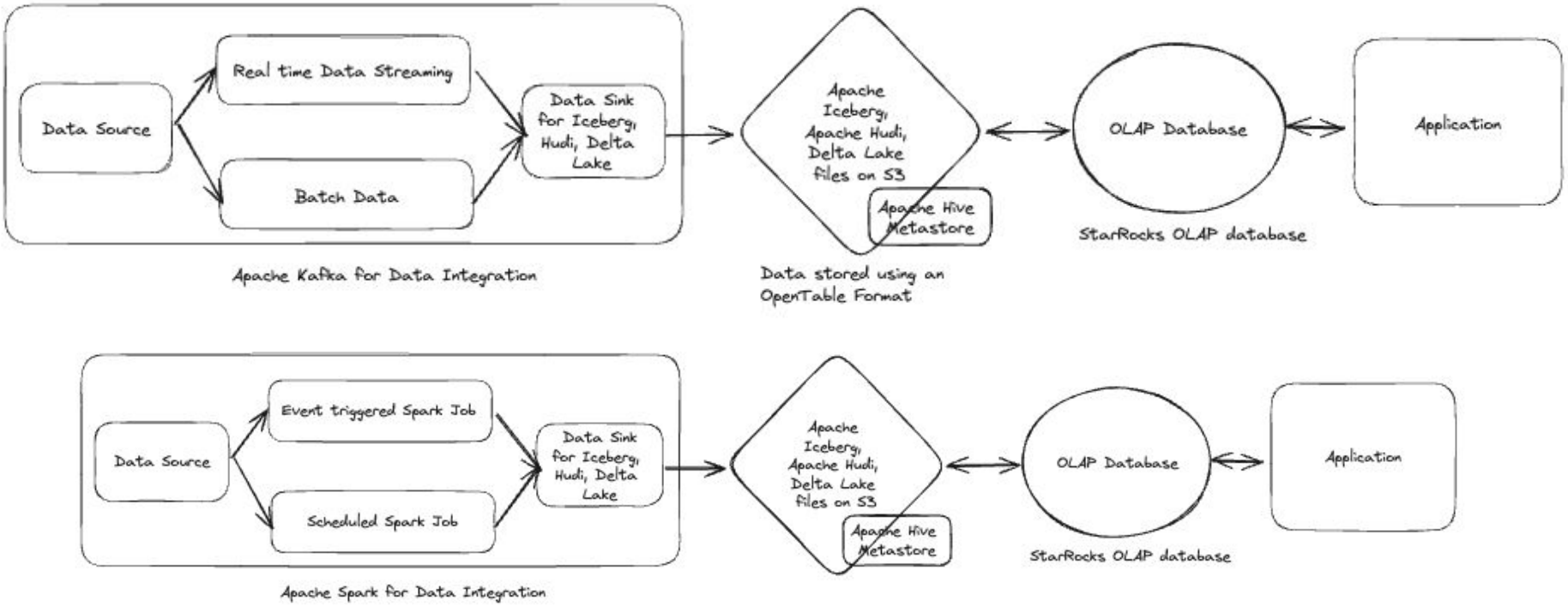
- This presentation is using images from Flaticon.com

Architectural Patterns and Best Practices

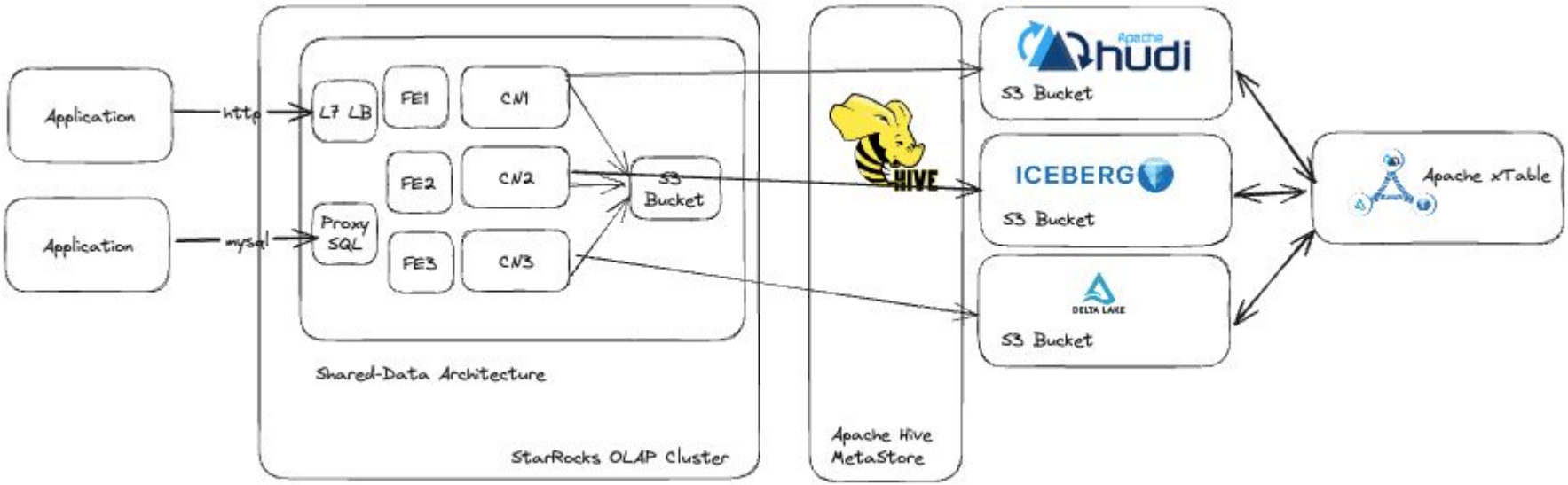
Kappa and Lambda Architecture with StarRocks + Apache Kafka



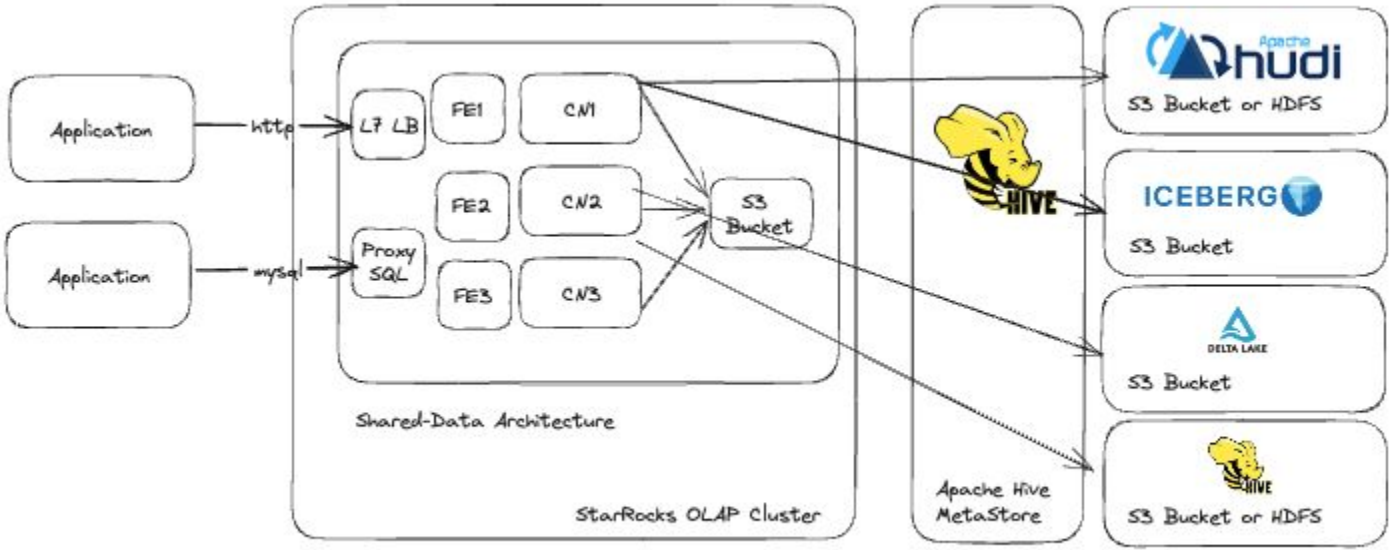
Kappa and Lambda Architecture with Open Lakehouse



Open Data Lakehouse with Apache xTable



Best Practices



Use StarRocks to query data in data lake. Deploy using the k8s operator.

Near real time replication from OLTP through Debezium or Kafka Sink Connect for Iceberg, Hudi, Delta Lake to write data directly into data lake.

For one time migrations or ELT, check out Sling Data.