StarRocks @ Grab

Presented by Gable Heng & Huong Vuong

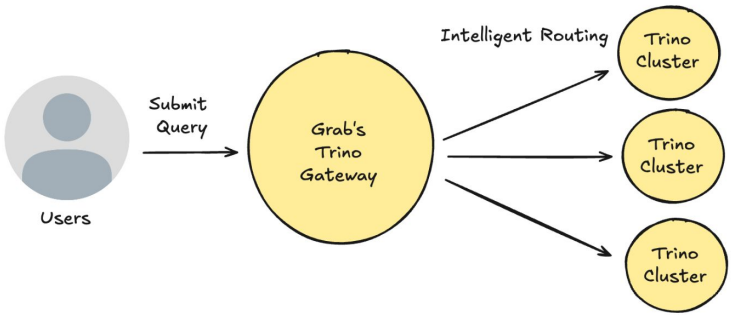
13th Mar 2025 | Version 1

Agenda

1. The state of interactive queries at Grab
2. Challenges faced regarding queries for BI and Dashboards
3. How StarRocks fits in to the picture
4. Using StarRocks to build Iris - A Spark Observability Product
5. Closing Notes + Q&A

Interactive Queries @ Grab

- Trino is currently Grab's main query engine for interactive queries.
- Queries are intelligently routed to Trino clusters.
- Routing is based on query analysis and Trino cluster health.
- Caching cannot be assumed due to routing and auto-scaling.
- Routing similar queries to one cluster to leverage caching increases risk of cluster issue.

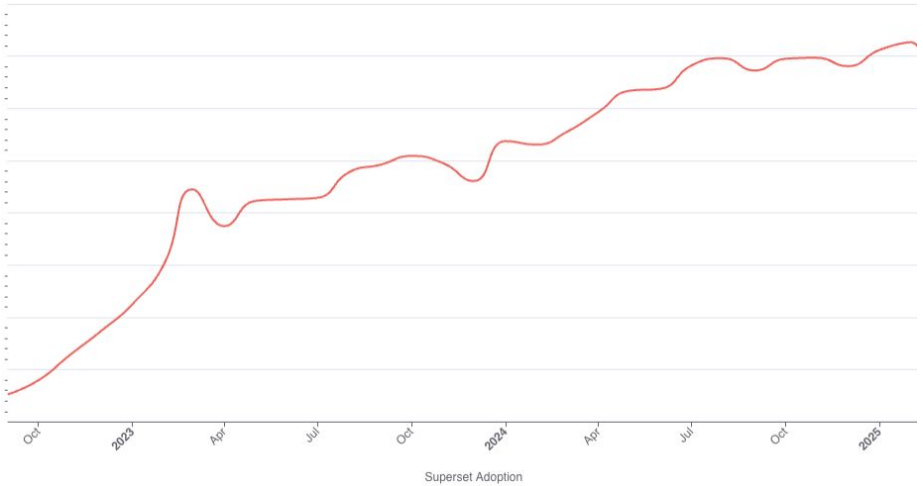


BI and Dashboarding

Introduction

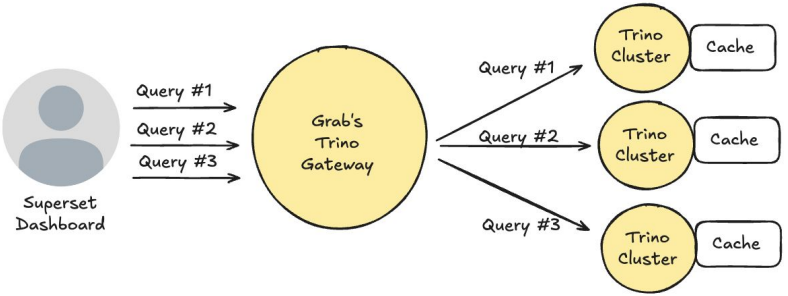
Main tools used are Superset and PowerBI

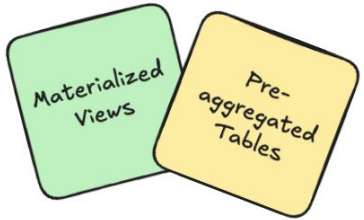
Introduction of web based tools such as Superset has significantly lowered the barrier to entry for users to unlock insights for their data



Challenge: Improving Query Performance

- Users build multiple charts on the same dataset.
- Multiple queries on the same dataset are sent to multiple clusters, makes it difficult to leverage caching.
- While query patterns are known, we are unable to leverage them effectively while also making it transparent to the user.

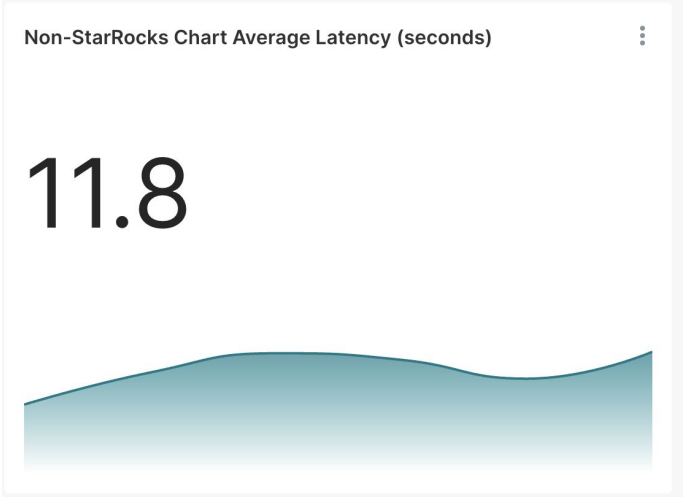


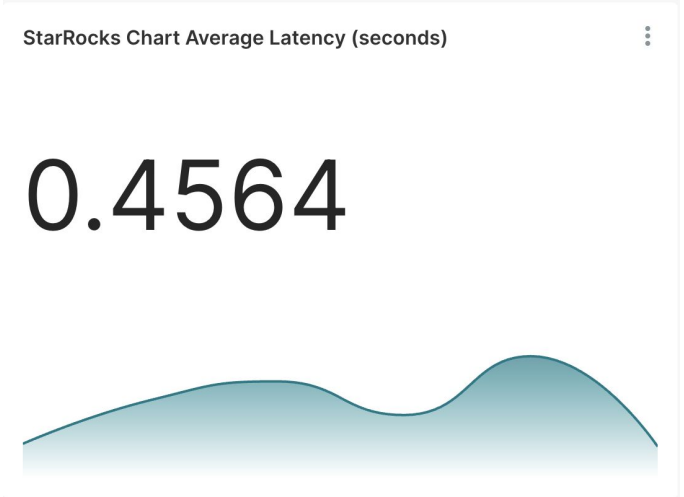


Advantages of using StarRocks

QUERY ACCELERATION	UNIFIED USER EXPERIENCE	DATA LAKE SUPPORT
Query cache with overlapping scanned partitions helps a lot with query performance due to similar query patterns Async Materialized Views are used to pre-aggregate and accelerate queries. Query Rewriting means users do not require context on which materialized views they need to use. Having multiple FE nodes helps maintain query performance during high concurrency scenarios	Trino SQL dialect support means that users can stick with just one dialect and not have to be careful of nuances of different dialects.	Support for querying data lake tables directly makes it easier to integrate with the rest of Grab's data lake. Data Lake Table or StarRocks Table is made transparent to the user. Provides a convenient fallback mechanism in the event of any downtime

Some Comparisons



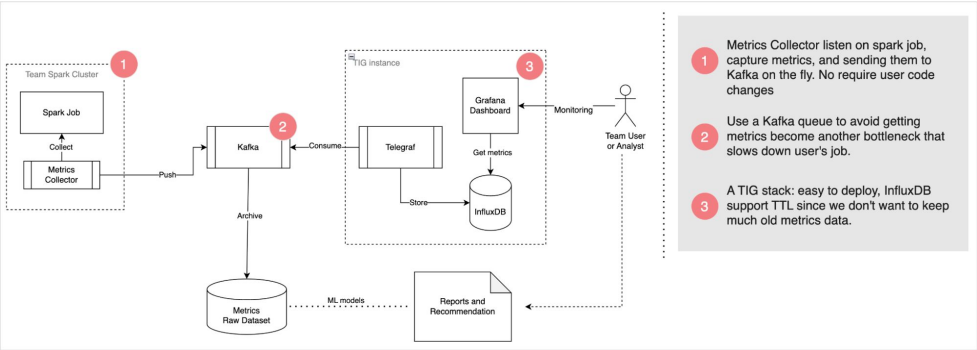


*This is a comparison of all charts currently in Superset and not a before-after comparison. Only the most frequently used charts are materialized in StarRocks for query acceleration.

Using StarRocks to Build Iris

- Iris was developed to provide in-depth observability for Spark jobs. Collects and analyzes metrics and metadata at the job level.
- Provides insights into resource usage, performance, and query patterns.
- Addresses critical gap by making Spark metrics accessible through a tabular dataset.
- Real-time performance metrics at the Spark application level.
- New Iris architecture integrates StarRocks.

Iris: Challenges and StarRocks Solution



TIG* stack: Poor metadata and join support, designed for numerical time series data.

TIG stack: Query performance issues with high cardinality data.

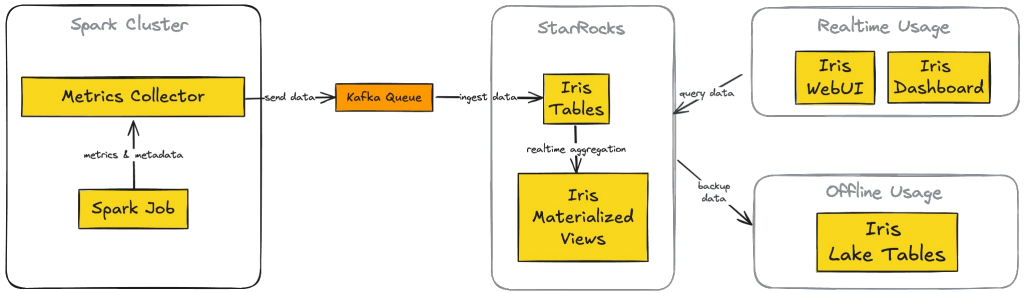
TIG stack: Limited SQL compatibility, requires use of Flux query language.

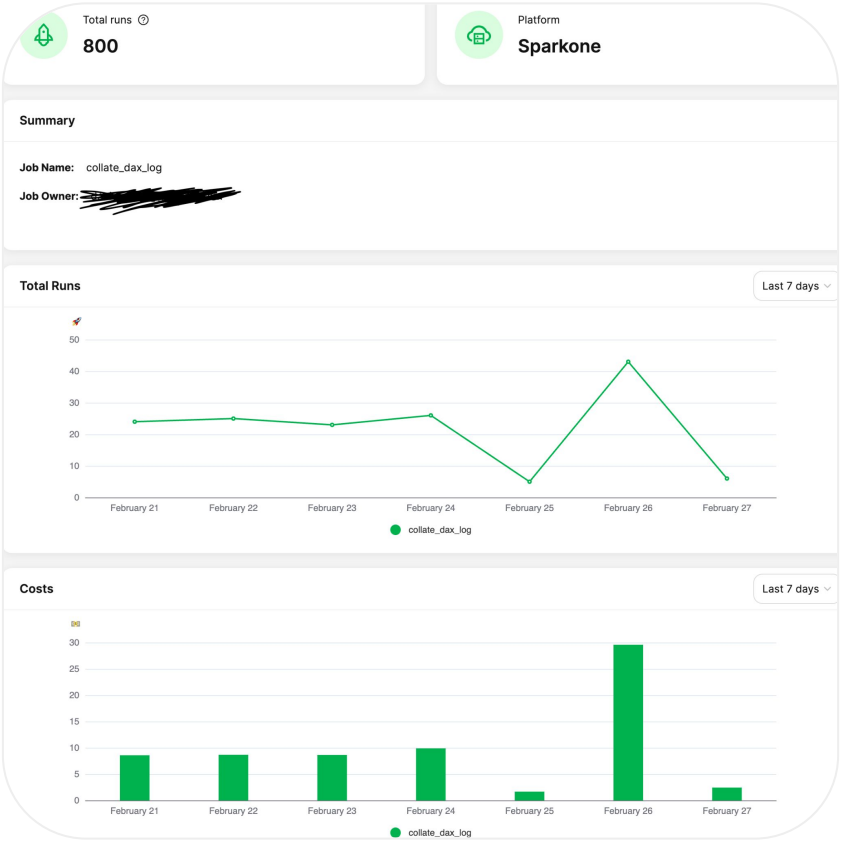
**TIG: telegraf, influxdb,grafana*

StarRocks: Efficient handling of both time-series and string-type metadata.

StarRocks: Optimized for high cardinality data.

StarRocks: Full MySQL-compatible SQL support.





Iris: Key Improvements from StarRocks

- Removed complexity of maintaining and managing the TIG stack
- Unblocked case to build Iris web application due to ease of connecting StarRocks using MySQL query
- Materialized view feature unblocked aggregation data and simplified management
- Dynamic partitioning and Time to Live (TTL) features remove overhead on engineering work
- Low latency data ingestion and aggregation improves near real-time data availability

Some challenges faced in StarRocks

- Ensuring compatibility with GrabTrino and StarRocks
- Rewriting queries to take advantage of query acceleration via Async MVs.
- Dealing with noisy neighbor effects as workers do both ingestion and querying.

Next Steps

- Expanding scope to serve more metrics and more users

- Exploring more potential use cases with StarRocks e.g real-time ingestion

Summary

1. StarRocks compatibility with Trino and Data Lake helps improve query performance while reducing learning curve and context required.
2. Features such as Async Materialized Views and multiple layers of caching help improve query performance while remaining transparent to the user.
3. StarRocks helps enable new data lake adjacent use cases for Grab such as Iris