Replicating data between transactional databases and ClickHouse®

Altinity

OSA Con 2024, November 21 2024

Who we are

Arnaud Adant

Database Team Lead

Jump Trading

Kanthi Subramanian

Open source contributor/Data Engineer/

Replicating data between transactional databases and CH

- Introduction
- Requirements
- Design choices
- Implementation
- Demo (5-10 min)
- Questions (5 min)

What is a transactional database ?

- Row based (tables, rows, …)
- B-tree data structure
- Speak SQL
- ACID
- Transactions
- MySQL, Postgres, … SQL Server, Oracle, …

What is ClickHouse ?

- An Open Source columnar database
- MergeTree data structure
- Created in 2009
- Realtime Analytics
- VLDB 2024 - ClickHouse: Lightning Fast Analytics for Everyone
- Over 2000 contributors

What is replication ?

- Data synchronization between a primary and replica(s)
- Homogeneous
- Heterogeneous
- Logical replication
- Log based
- Change Data Capture

Why replication ?

- Migration : system A to B
- Continuous synchronization
- Fault tolerance
- Disaster recovery
- Very useful building block

Use case for replicating to ClickHouse

- Hybrid Transactional and Analytics Processing
- Alternative to proprietary solutions
- Real time Analytics
- Running out of space in MySQL / Postgres
- Data archival
- History tables

Requirements

- No Data Loss
- Low Latency
- Exactly Once Delivery
- Operational Simplicity
- Fault Tolerance / HA
- Fully Open Source

Design choices

- Keep It Simple Stupid
  - A docker container, simple yaml config

- Standing on the shoulders of giants :
  - MySQL binary logs, ANTLR, PG Logical replication

- Do not re-invent the wheel :
  - Debezium for Change Data Capture

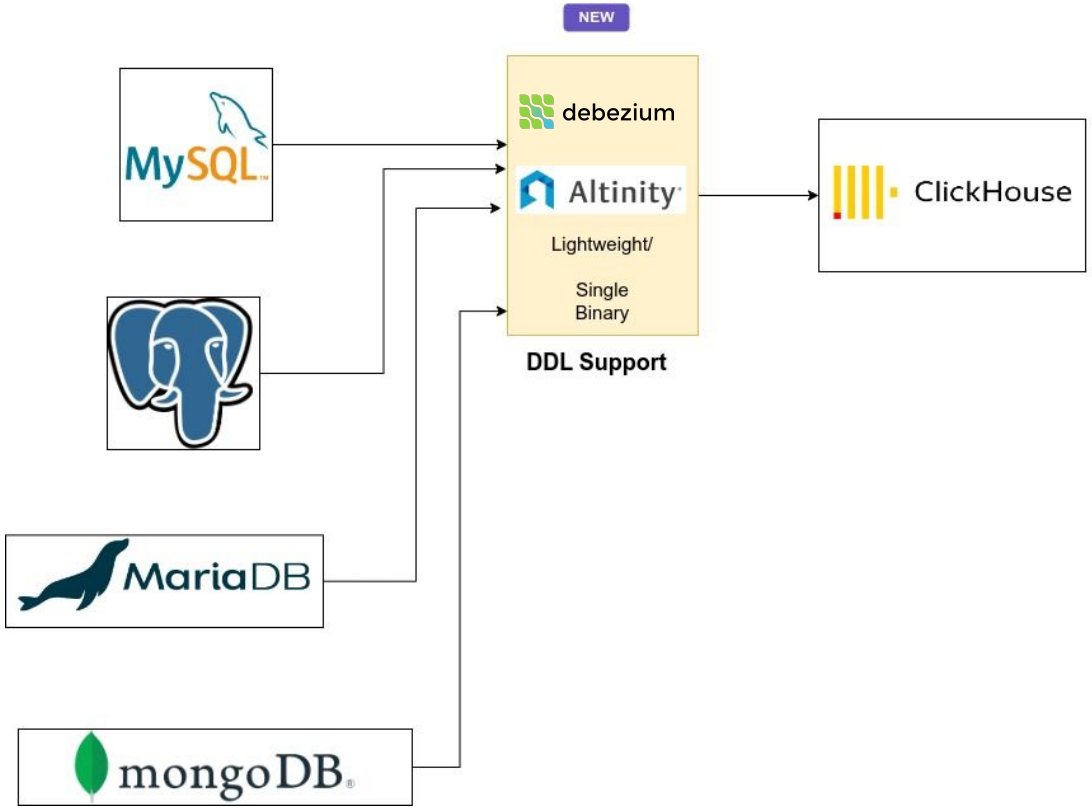
Design choices in ClickHouse (1/2)

- One to One Table mapping
- Primary key
- Data Type Mapping
- ReplacingMergeTree engine, aka RMT
- “the same queries should return the same results”
- Full DDL Support
- Timezones

Design choices in ClickHouse (1/2)

- Replication state stored in ClickHouse
- MySQL replication like behavior (stop / start / status)
- Retry on failure, both DML and DDL
- Replication filters
- Checksums
- Efficient dumpers and loaders

Architecture



Implementation : Sink Connector Lightweight

- One container
- Docker compose
- Java and Debezium based
- Multi-threaded Applier
- Eventually consistent
- Low Latency

Table Migration


```
CREATE TABLE `orders` (
`orderNumber` int NOT NULL,
`orderDate` date NOT NULL,
`requiredDate` date NOT NULL,
`shippedDate` date DEFAULT NULL,
`status` varchar(15) NOT NULL,
`comments` text,
`customerNumber` int NOT NULL,

PRIMARY KEY (`orderNumber`),
KEY `customerNumber` (`customerNumber`) )
ENGINE=InnoDB DEFAULT CHARSET=latin1

PARTITION BY RANGE COLUMNS(orderDate)
(
PARTITION p20201231 VALUES LESS THAN ('2021-01-01'),
PARTITION p20211230 VALUES LESS THAN ('2021-12-31')
)
```

```
CREATE TABLE test.orders (
`orderNumber` Int32,
`orderDate` Date32,
`requiredDate` Date32,
`shippedDate` Nullable(Date32),
`status` String,
`comments` Nullable(String),
`customerNumber` Int32,

`_version` UInt64,
`is_deleted` UInt8 )

ENGINE = ReplacingMergeTree( _version, is_deleted)
ORDER BY orderNumber
PARTITION BY orderDate
SETTINGS index_granularity = 8192
```

Altinity

Demo

Altinity Sink Connector for ClickHouse

https://github.com/Altinity/clickhouse-sink-connector

GitHub

slack

Questions

Appendix

Roadmap

- History tables
- Override column Schema (custom mapping of data types)
- Configuration builder for non-expert users.
- Support for Transactions.
- Support for more source databases (MongoDB, Cassandra, SQL Server, Oracle)

Features

Features

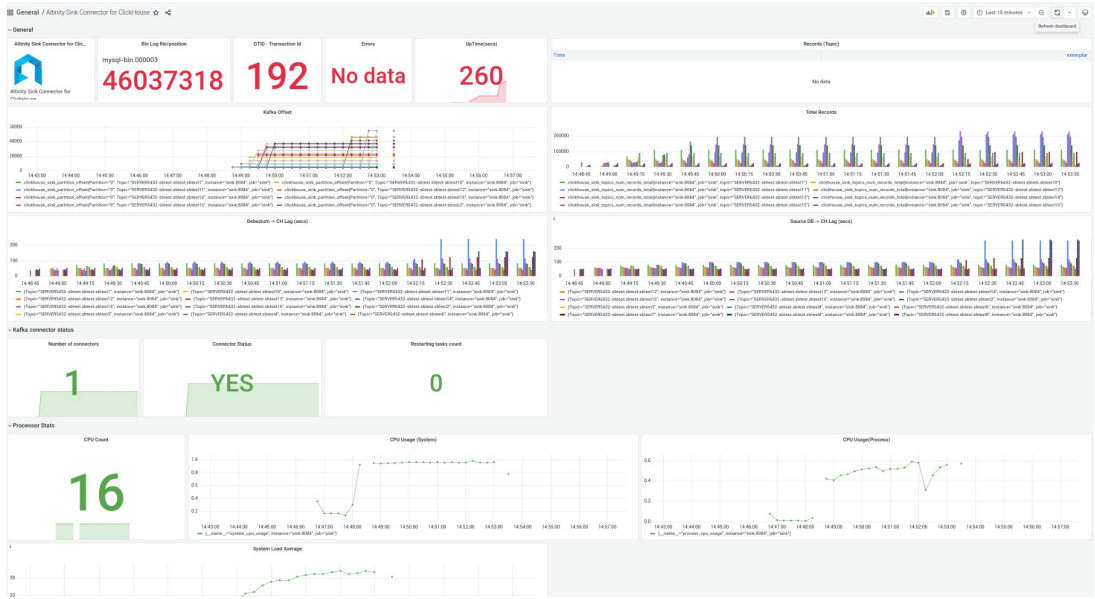
Feature	Description
Single Binary	No additional dependencies or infrastructure required
Exactly Once Processing	Offsets are committed to ClickHouse after the messages are written to ClickHouse
Supported Databases	MySQL, MariaDB, PostgreSQL, MongoDB(Experimental)
Supported ClickHouse Versions	24.8 and above
Clickhouse Tables Types	ReplacingMergeTree, MergeTree, ReplicatedReplacingMergeTree
Replication Start positioning	Using sink-connector-client to start replication from a specific offset or LSN(MySQL Binlog Position, PostgreSQL LSN)
Supported Datatypes	Refer Datatypes
Initial Data load	Scripts to perform initial data load (MySQL)
Fault Tolerance	Sink Connector Client to continue replication from the last committed offset/LSN in case of a failure
Update, Delete	Supported with ReplacingMergeTree
Monitoring	Prometheus Metrics, Grafana Dashboard
Schema Evolution	DDL support for MYSQL.
Deployment Models	Docker Compose, Java JAR file, Kubernetes
Start, Stop, Pause, Resume Replication	Supported using sink-connector-client
Filter sources databases, tables, columns	Supported using debezium configuration.
Map source databases to different ClickHouse databases	Database name overrides supported.
Column name overrides	Planned
MySQL extensive DDL support	Full list of DDL(sink-connector-lightweight/docs/mysql-ddl-support.md)
Replication Lag Monitoring	Grafana Dashboard and view to monitor lag
Batch inserts to ClickHouse	Configurable batch size/thread pool size to achieve high throughput/low latency
MySQL Generated/Alias/Materialized Columns	Supported
Auto create tables	Tables are automatically created in ClickHouse based on the source table structure.

Comparison

Feature	Altinity Sink Connector (Lightweight, Single Binary)	Airbyte	ClickHouse `mysql` Table Engine	Custom Python Script with ClickHouse Connect
Replication Type	Real-time CDC	Batch (Scheduled)	Direct Query	Batch or Scheduled
Data Freshness	Near real-time	Configurable (e.g., hourly)	Near real-time (with latency)	Configurable
Schema Change Handling	Full support(MySQL), Partial(PostgreSQL)	Manual schema refresh required	No automatic schema sync	Manual intervention needed
Complexity	Low to Medium (single binary setup)	Moderate	Low	High (requires coding and scheduling)
Ease of Setup	Easy (standalone binary, no Kafka needed)	Easy	Very easy	Complex (custom coding)
Maintenance	Low to Moderate (single binary process)	Low	Low	High
Initial Sync Support	Yes	Yes	Not applicable (direct query)	Yes
Transformation Capabilities	Limited	Basic (Airbyte transformations)	No	Full control (custom code)
Cost	Free or license-based	Free (Open-source)	Free (built-in to ClickHouse)	Free (but may require custom infrastructure)
Suitability for High Volume	High	Medium	Medium	Medium to Low
Additional Infrastructure	None	None	None	Optional (scheduling tools like Airflow)
Data Accuracy	High (real-time CDC)	Medium (depends on sync frequency)	Medium	High
Ideal Use Case	Low-latency, real-time replication without Kafka	Batch syncs, easy setup	Simple queries without replication	Custom, flexible ETL

Monitoring

Monitor Lag, DDL and CPU/Memory Usage using Grafana Dashboard.
Start/Stop replication and monitor lag using sink connector CLI.



Altinity

Development

- Actively developed with feedback from Customers with serious production workloads

- Contributors with expertise in ClickHouse, JDBC and Debezium.

Altinity

Performance

- Faster than MySQL replication
- Configurable ClickHouse writer thread pool
- Configurable Queue size
- Configurable batch size

Performance