Migrating 100 Billion rows of financial data

- Existing Setup
- Challenges with Postgres and need for migration
- Migration Strategy and the nuances
- Single node Postgres to Multi-node Clickhouse(s)
- Benefits

WhoAmI

- Ritesh Shrivastav
- Software Engineer at Zerodha
- Over 7 years of building and scaling Console
- @johnnybravo-xyz <ritesh@shrivastav.xyz>

Existing Setup

- Console stores and processes reports for over 10 million clients
- Over 100 billion rows across 8 tables using ~40TB of storage
- Single node Postgres that serves live traffic
- One more supplementary Postgres to aid ETL-pipeline
- This talk is about migrating data from this supplementary Postgres.

Challenges with Postgres

- Pushed single-node Postgres to its limits
- Hardware requirements for maintaining these nodes are prohibitively expensive
- Storage was also becoming a bottleneck.
- ETL pipelines became choke point
  - Ingesting data took 3.5 hours on average
  - Low throughput while ingesting data into multiple tables parallely

Why Clickhouse?

- SQL support with more goodies
- Queries are lightning fast
- Data ingestion too is lighning fast
- Data Compression
- Materialized Views
- Easy to scale on commodity server(s)
- Cons?
  - Update/Deletes have to be planned well
  - Data merge timelines are non-definite

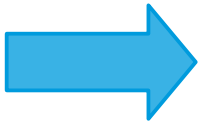
Migration

- Make sure existing queries can be changed/adapted to CH’s SQL
- Finding the right schema
  - Table Engines
  - Data Types
  - Compression Codecs

Sample Table Schema

- Postgres

```
CREATE TABLE orders (
    user_id              varchar
    order_date           date
    order_punch_time     datetime
    order_exec_time      datetime
    trade_type           varchar
    quantity             decimal
    price                decimal
    exchange             varchar
    symbol               varchar
    secondary_id         varchar
    order_id             varchar
)
PARTITION BY RANGE(order_date)
```



- Clickhouse

```
CREATE TABLE orders (
    user_id              String
    order_date           Date
    order_time           Array(DateTime)
    trade_type           LowCardinality(String)
    qty_price            Array(Decimal)
    exch_symbol          Array(String)
    order_ids            Array(String)
)
ENGINE = MergeTree
PARTITION BY toYYYYMM(order_date)
ORDER BY (order_date, user_id)
SETTINGS index_granularity = 365
```

Migration

- Convert data into CSVs with format that fits the new schema
- Move them via block storages or use HTTP interface

```
$ psql> \COPY (
    SELECT
        user_id, order_date,
        trade_type,
        JSON_BUILD_ARRAY(order_punch_time, order_exec_time) AS order_time,
        JSON_BUILD_ARRAY(price, quantity) AS qty_price
        JSON_BUILD_ARRAY(exchange, symbol) AS exch_symbol,
        JSON_BUILD_ARRAY(secondary_id, order_id) AS order_ids
    FROM orders
    WHERE
        order_date >= '2017-01-01'
        AND order_date <= '2017-12-31'
 ) TO '/tmp/ch-orders/partition-1-2017.csv' CSV HEADER
```

```
$ cat /tmp/ch-orders/partition-1-2017.csv | clickhouse-client -d console --query “INSERT INTO orders FORMAT CSVWithNames”
```

Sharding Data into multiple nodes

- Don’t repeat mistake of single-node setup like Postgres
- Clickhouse scales linearly so, its easy to deploy multiple instances.
- Physically shard data into multiple nodes
- Deal with multiple shards in app logic than to get into menace of zookeeper and all.
  - CH has improved with CH-Keeper and we plan to take a look at it again in future

Benefits

- We have six EC2 instances for the price of one huge EC2 instance used for single-node Postgres
- 50% savings on storage, can be improved further with use of codecs
- Data ingestion is 10x faster
- Extracting data in bulk for ETL-pipeline takes 10 mins compared to 140 mins from previous setup
  - Ingest data in parallel to multiple tables