Welcome to the ClickHouse meetup!

India Bangalore

2024

ClickHouse

ClickHouse

A Quick Overview

2024

ClickHouse

Speakers

Johnny Mirza

Solutions Architect

What is ClickHouse?

Your (soon-to-be) favorite database!

Open source **column-oriented** **distributed** **OLAP** **database**

Since 2009	Best for aggregations	Replication	Analytics use cases
31,000+ GitHub stars	Files per column	Sharding	Aggregations
1300+ contributors	Sorting and indexing	Multi-master	Visualization
500+ releases	Background merges	Cross-region	Mostly immutable data

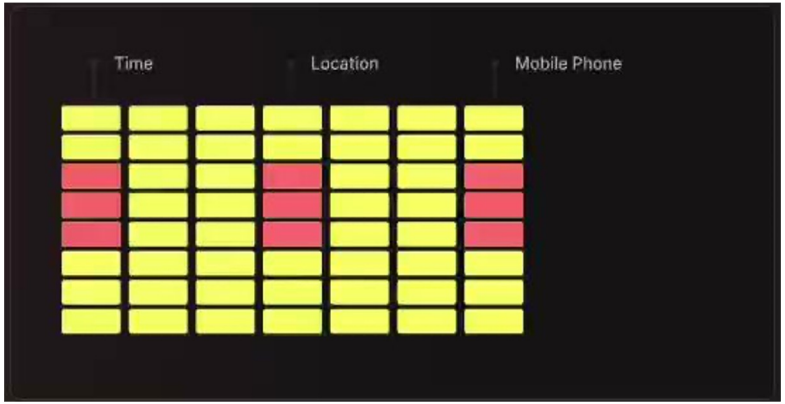
What is OLAP?

Column-oriented databases are better suited to OLAP scenarios. They are at least 100x faster in processing most queries. ClickHouse uses all available system resources to their full potential to process each analytical query as fast as possible.

Row-Oriented

Data is stored in rows, with all the values related to a row physically stored next to each other.

Column-oriented



In ClickHouse, data is stored in columns, with values from the same columns stored together.

Key Features

Some of the cool things ClickHouse can do

1 Speaks SQL

Most SQL-compatible UIs, editors, applications, frameworks will just work!

2 Lots of writes

Up to several million writes per second - per server.

3 Distributed

Replicated and sharded, largest known cluster is 4000 servers.

4 Highly efficient storage

Lots of encoding and compression options - e.g. 20x from uncompressed CSV.

5 Very fast queries

Scan and process even billions of rows per second and use vectorized query execution.

6 Joins and lookups

Allows separating fact and dimension tables in a star schema.

Database for Interactive Experiences

Make any data fast

User-facing Analytics	Internal Analytics	Observability
Spotify	Walmart	CLOUDFLARE
Bloomberg	Deutsche Bank	GitLab
twilio	lyft	NETFLIX
CONTENTSQUARE	PostHog	COMCAST

Benchmarks

- Our own at https://clickhouse.com/benchmark/dbms/
  - 6x faster than Vertica
  - 25x faster than Greenplum
  - 50x faster than TimescaleDB
- Others, see https://github.com/ClickHouse/ClickHouse/issues/22398
  e.g.:
  - 3-5x faster than RedShift (sometimes 50x)
  - 5-6x faster than Elasticsearch
  - 6-10x faster than Druid
- Storage
  - Ebay: 90% less hardware than Druid
  - Contentsquare: 6x more data than Elasticsearch

Creating tables

ClickHouse SQL Basics

```
CREATE TABLE customers
(
    name      String,
    age       UInt8,
    address   Array(String),
    city      LowCardinality(String),
    created   DateTime,
    type      Enum8(...),
    attr      Map(String, Boolean)
)
ENGINE = MergeTree
ORDER BY (city, name, type)
```

- Other engines:
  - ReplacingMergeTree
  - CollapsingMergeTree
  - AggregatingMergeTree
- Integration engines:
  - Kafka, RabbitMQ
  - MySQL, PostgreSQL, MongoDB
  - JDBC, ODBC
  - S3, HDFS
  - EmbeddedRocksDB
- Adding Replicating- in front makes an engine replicate data (e.g. ReplicatingMergeTree)

Inserting directly

ClickHouse SQL Basics

```
INSERT INTO people VALUES ('Obi-Wan Kenobi', 57, ...) ('Yoda', 900, ...)
(...)
```

- Batching is important! Either batch yourself or turn on asynchronous inserts:

```
SET async_insert = true
INSERT INTO people VALUES ('Obi-Wan Kenobi', 57, ...)
INSERT INTO people VALUES ('Yoda', 900, ...)
INSERT INTO people VALUES (...)
```


Inserting from external sources

ClickHouse SQL Basics

- ClickHouse has many built-in table functions to read external data:

```
INSERT INTO table SELECT * FROM s3(...)
INSERT INTO table SELECT * FROM file(...)
INSERT INTO table SELECT * FROM url(...)
INSERT INTO table SELECT * FROM mysql(...)
INSERT INTO table SELECT * FROM postgresql(...)
INSERT INTO table SELECT * FROM jdbc(...)

And more!
```

- Read the docs at clickhouse.com/docs!

Reading data

ClickHouse SQL Basics

- Don’t do this! (at least not too often)

```
SELECT * FROM table
```

- And also not this! (at least not too often)

```
SELECT ... FROM table WHERE row_id = '<uuid>'
```

- The majority of your queries should be:

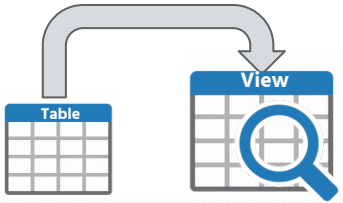
```
SELECT avg(something) FROM table WHERE toYear(timestamp) = 2022
```


Views

ClickHouse SQL Basics

- Saving a query as a view (no data movement):

```
CREATE VIEW view AS SELECT ... FROM table ...
```



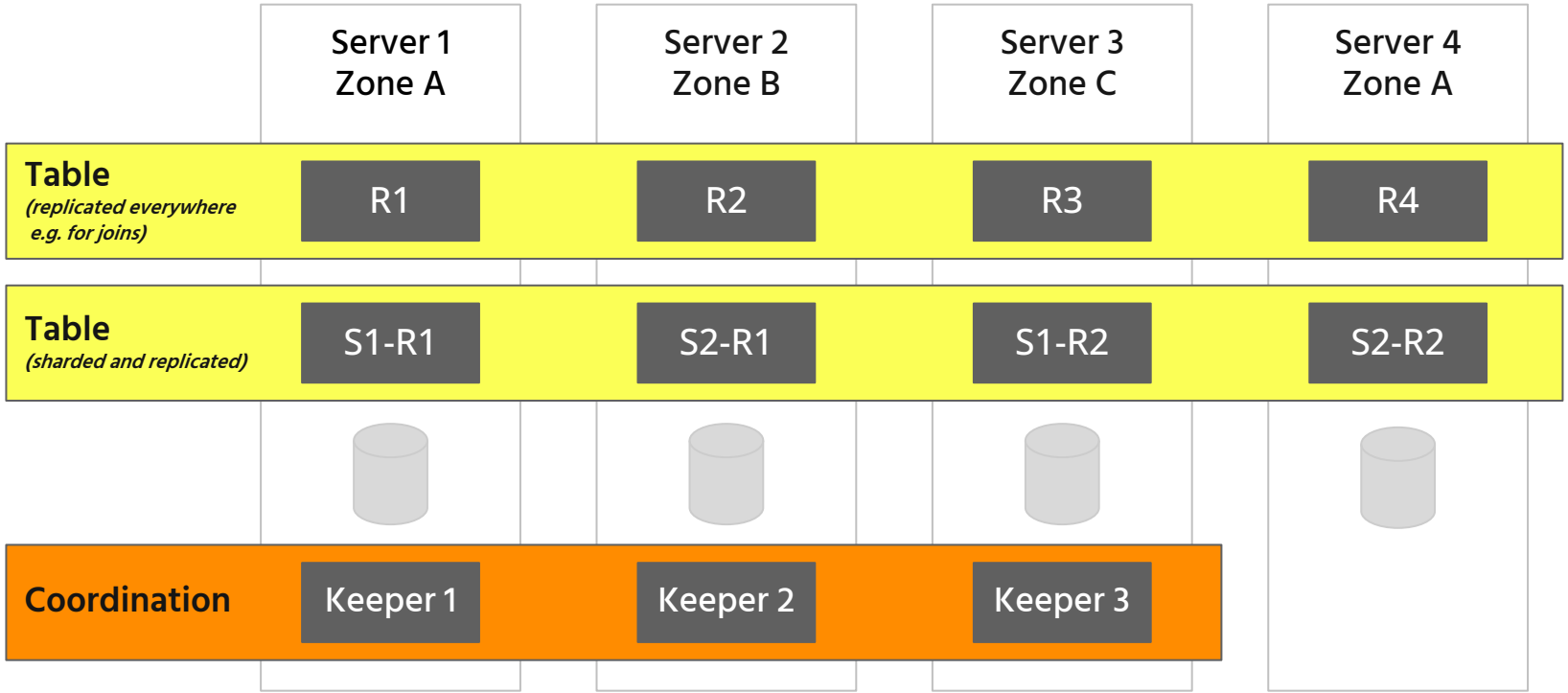
- Continuously processing new data from a table into another table:

```
CREATE MATERIALIZED VIEW view AS SELECT avg(...) FROM table GROUP BY ...
```

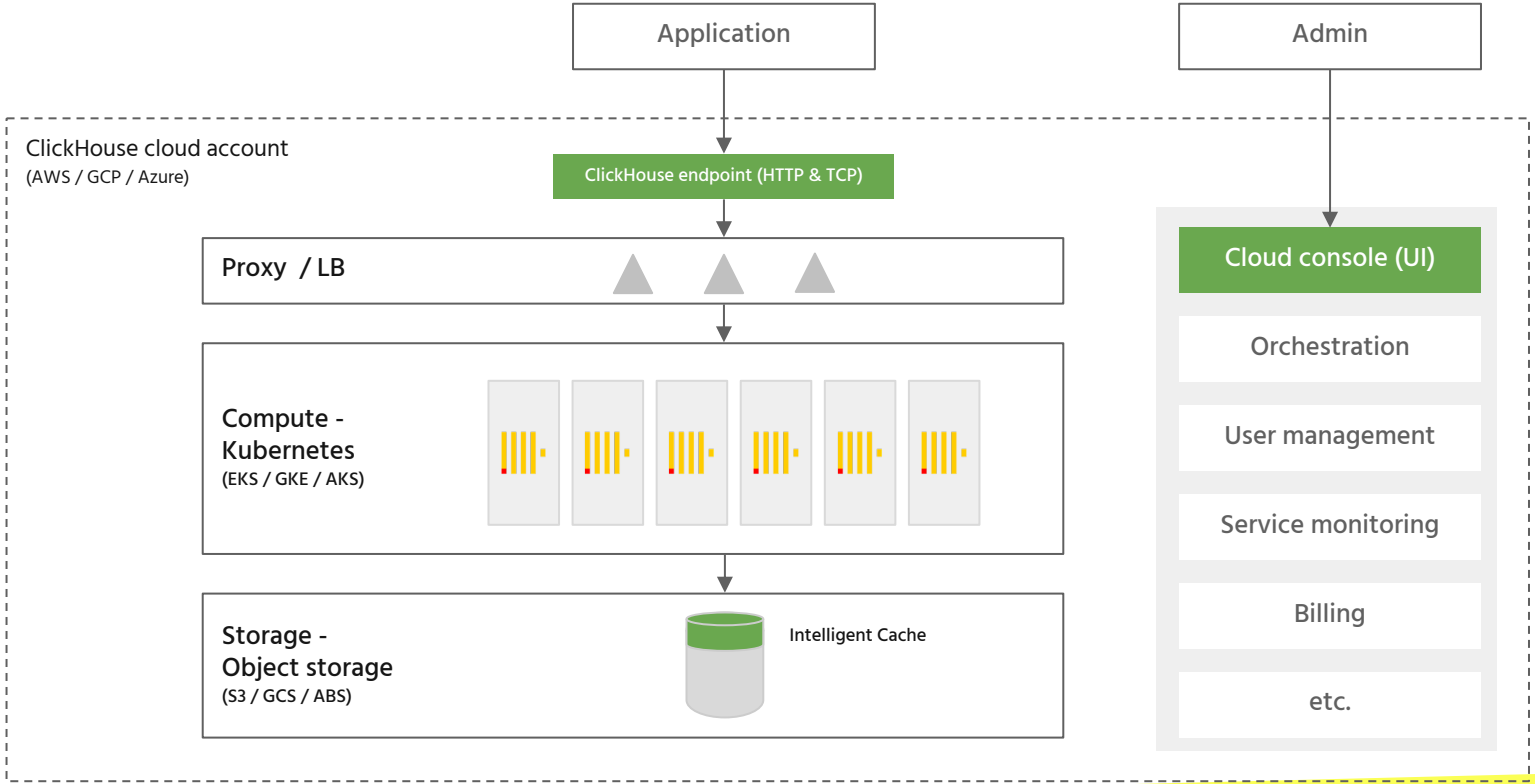

Ecosystem and Integrations

- Interface
  - HTTP, native TCP, CLI
  - JDBC, ODBC
  - Client libraries: Go, Python, Javascript, etc.
- Data ingest
  - Kafka, S3, HDFS, RabbitMQ (native)
  - MySQL, PostgreSQL, MongoDB, Redis (native)
  - Vector (logs and metrics)
  - Airbyte (ELT)
  - JDBC, ODBC (Spark, Flink, etc.)
- Query editor
  - Open source: DBeaver, VS Studio
  - Proprietary: DataGrip
- Visualization
  - Open source: Grafana, Superset, Metabase
  - Proprietary: Tableau, Power BI

Architecture



ClickHouse Cloud Architecture



Let’s do a quiz!

ClickHouse Quiz

What is ClickHouse?

A. OLTP

B. Key-Value

C. Document

D. OLAP

Pub Quiz

Is ClickHouse...

A **Fully consistent**

B **Eventually consistent**

Pub Quiz

Is ClickHouse...

A Schema

B Schemaless

20

Pub Quiz

Is ClickHouse...

21

Pub Quiz

Why is ClickHouse fast?

A. C++

B. Sorting

C. SIMD

D. Lots of love

22

Pub Quiz

How much data can you store in ClickHouse?

A. Gigabytes

B. Terabytes

C. Petabytes

D. Exabytes