ClickHouse

A Technical Overview

2024

ClickHouse

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

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	Disney+

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 events
(
    timestamp DateTime,
    tenant    LowCardinality(String),
    type      Enum8(...),
    value     Float64,
    attr      Array(String),
    flags     Map(String, Boolean)
)
ENGINE = MergeTree
ORDER BY (tenant, timestamp)
```

- 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 id = '<uuid>'
```

- The majority of your queries should be:

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

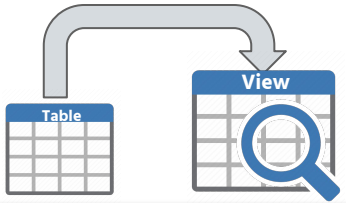

Views

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:

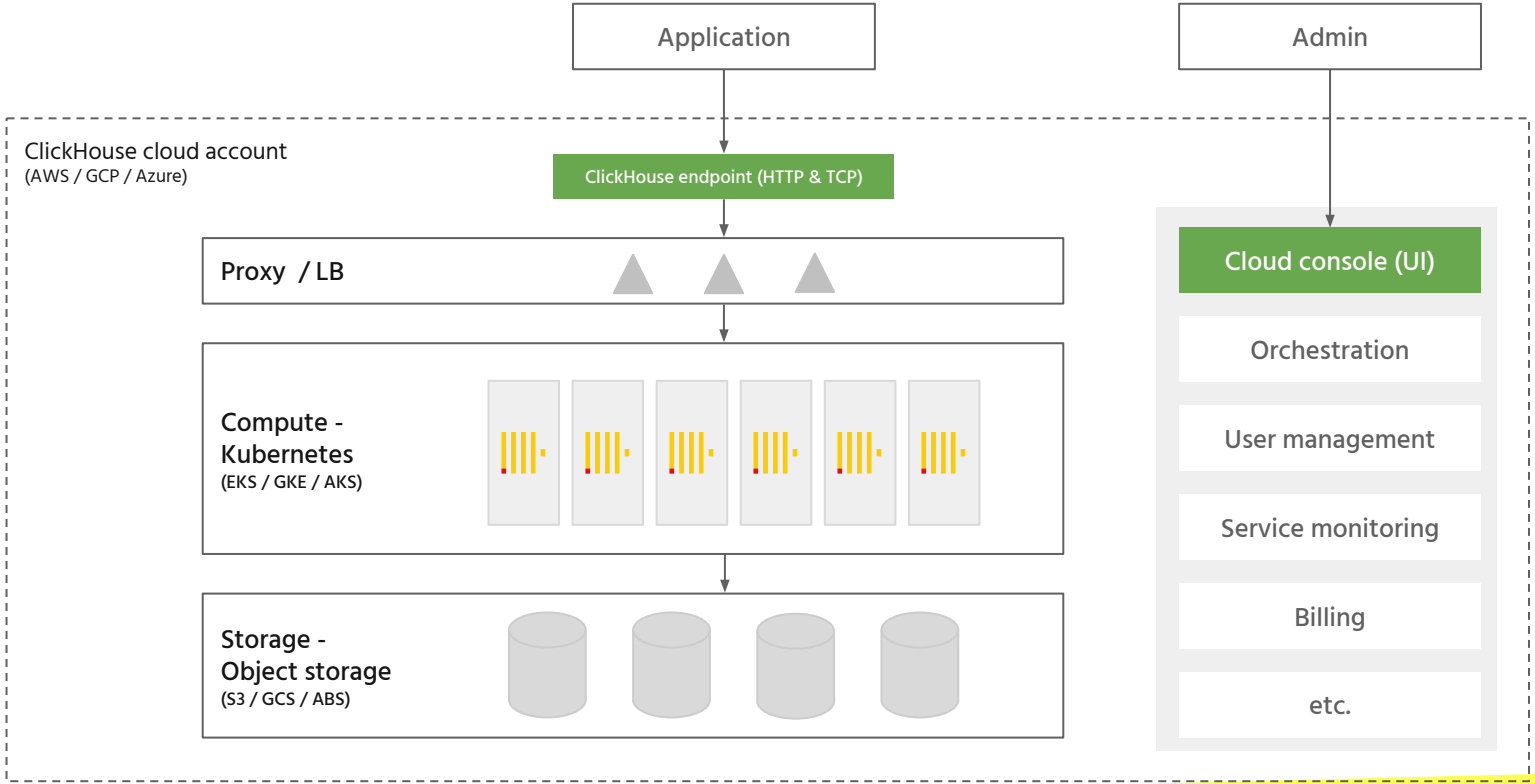



```
CREATE MATERIALIZED VIEW view AS SELECT avg(...) FROM table GROUP BY ...
```

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

ClickHouse Cloud Architecture



For more, go to clickhouse.com/learn

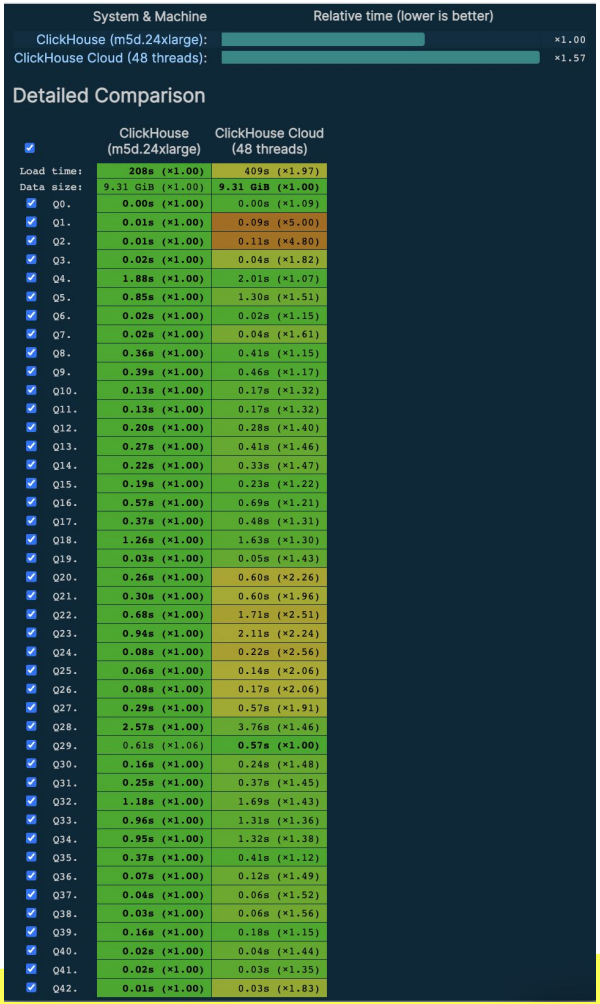
Let’s take a look!

ClickHouse Cloud Benchmark

Are queries still fast on top of object store?

https://benchmark.clickhouse.com/

- Object storage but similar bandwidth to local storage, and query concurrency is not affected
- Object storage has slower access latency, compared to attached SSD
- There is some slowdown, but only for short queries
- ClickHouse Cloud multi-layer cache ensures fast analytical query results despite slower access latency of the underlying store
- The fact that data is not replicated can be used to speed up certain queries



Deployment

- Deploy anywhere
  - Any cloud provider or self-managed
  - Bare metal, VM, or Docker/Kubernetes
  - ClickHouse Cloud (serverless)
- Cross-datacenter
  - Multiple availability zones is fine
  - DCs or cloud regions on the same continent is fine (e.g. us-west and us-east)
  - Better not go across oceans
- Storage
  - SSD - fast and expensive
  - HDD - slower and cheaper
- Scaling
  - Bigger servers first, then more servers (don’t be afraid of big servers!)

Architecture