ClickHouse Top Highlights H2 2022

Reliability

ClickHouse should always work.

Asynchronous Initialization Of Tables

If ZooKeeper is unavailable at server startup,
the **ReplicatedMergeTree** tables will start in read-only mode
and initialize asynchronously in background
as soon as ZooKeeper will be available.

— the same applicable for ClickHouse Keeper as well;

— especially useful for embedded ClickHouse Keeper;

Developers: Antonio Andelic. Since 22.9.

Retries On INSERT

Session expired. Table is in readonly mode ☹

Never again:

```
SET insert_keeper_max_retries = 10;
```

INSERT will survive restarts of ClickHouse Keeper or ZooKeeper
and reconnections.

Developer: Igor Nikonov. Since 22.11.

Faster Reconnection to Keeper

Table is in readonly mode 😡

Was: around one minute for reconnection, for no reason.

Now: milliseconds 🥲

Developer: Raul Marin. Since 22.10.

No More "Too many parts"

DB::Exception: Too many parts (300). Merges are processing significantly slower than inserts ☹

Now: relaxing the "too many parts" threshold:

Allow large number of parts if they are large in average.

Developer: Alexey Milovidov. Since 22.10.

Flexible Memory Limits

Was: strict per-query limit, **max_memory_usage** = 10 GB by default.

Now: query can use the memory if it's available;
in case of memory shortage, the most "overcommitted" query is terminated with exception.

Developer: Dmitry Novik.

ClickHouse Keeper

No need to use ZooKeeper anymore!

Always use ClickHouse Keeper instead of ZooKeeper.

Developer: Alexander Sapin, Antonio Andelic.

SQL Compatibility

ClickHouse should support everything you expect.

Window Functions Inside Expressions

```
SELECT
    y::String || '.'
        || (y < toYear(today()) - 2000 - 1 ? '*' : m::String) AS Version,
    (n <= 3 OR (is_lts AND lts_n <= 2)) ? '✔' : '❌' AS Supported
FROM (
    SELECT y, m,
        count() OVER (ORDER BY y DESC, m DESC) AS n,
        m IN (3, 8) AS is_lts,
        countIf(is_lts) OVER (ORDER BY y DESC, m DESC) AS lts_n
    FROM (
        WITH extractGroups(version, 'v(\d+)\.(\d+)') AS v,
             v[1]::INT AS y, v[2]::INT AS m
        SELECT y, m
        FROM file('version_date.tsv', TSV, 'version String, date String')
        ORDER BY y DESC, m DESC
        LIMIT 1 BY y, m)
)
LIMIT 1 BY Version
FORMAT Markdown
```

Extended Range For Date&Time

Supported range for **DateTime64** and **Date32** data types:

Was: **1925–2283.**

Now: **1900–2300.**

Uses proleptic Gregorian calendar.

Motivation: store dates of birth of the customers.

Developer: Roman Vasin. Since 22.8.

DELETE Query

```
SET mutations_sync = 1;
ALTER TABLE hits
DELETE WHERE Title LIKE '%Mongo%';
```

— 205 sec (for a table with 100 million records).

```
DELETE FROM hits
WHERE Title LIKE '%Mongo%';
```

— ??? sec.

Developers: Alexander Gololobov, Jianmei Zhang.

GROUPING Function

Used with ROLLUP, CUBE or GROUPING SETS.

To distinguish different sets.

```
SELECT k, GROUPING(k) FROM table GROUP BY k WITH ROLLUP
```

Developer: Dmitriy Novik.

Non-Constant LIKE and match

```
SELECT DISTINCT repo_name, title
FROM github_events
WHERE title ILIKE (
  repo_name LIKE '%ClickHouse%' ? '%fast%' : '%slow%')
AND repo_name IN ('ClickHouse/ClickHouse', 'elastic/elasticsearch')
```

Now I can put LIKE inside LIKE and looks like you're going to like it.

Developer: Robert Schulze. Since 22.6.

Composite Time Intervals

Examples:

```
SELECT now() + INTERVAL 1 MONTH;
SELECT now() + INTERVAL '1 MONTH';
SELECT now() + INTERVAL 1 MONTH - INTERVAL 2 DAY;

SELECT now() + INTERVAL '1 MONTH -2 DAY';
SELECT now() + (INTERVAL 1 MONTH - INTERVAL 2 DAY);

SELECT INTERVAL '1 MONTH -2 DAY';
SELECT (INTERVAL 1 MONTH - INTERVAL 2 DAY);

SELECT INTERVAL '1 MONTH 1 MONTH';
```

Developer: Nikolai Degterinsky. Since 22.11.

Performance

ClickHouse never slows down!

Performance Optimizations

Improvement of ORDER BY, insert and merge in MergeTree,
and window functions.

```
SELECT WatchID FROM hits_100m_obfuscated ORDER BY Age
```

Before:
Elapsed: **4.154** sec. (24.07 million rows/s., 216.64 MB/s.)

After:
Elapsed: **0.482** sec. (207.47 million rows/s., 1.87 GB/s.)

Developer: Maksim Kita.

Performance Optimizations

Speed-up of SELECT with **FINAL** modifier.

It "simply" improves performance up to 4 times.

Especially for complex transforms like Collapsing and Replacing.

Developer: Nikita Taranov.

Performance Performance

Optimize **ORDER BY with LIMIT**.

Optimize **ORDER BY** with single column.

Optimize **INSERT** into MergeTree with composite ORDER key.

Optimize **dictGetChildren**, **dictGetDescendants**.

Optimize cleanup stage of queries with large **GROUP BY**.

Optimize **background CPU usage** of large number of tables.

Developer: Maksim Kita, Nikita Taranov.

More Performance

Optimize **COUNT(DISTINCT ...)** for low number of GROUP BY keys.

Optimize **GROUP BY** with CPU prefetcher.

Optimize **GROUP BY** with better block sizes.

Developer: Nikita Taranov.

New JOIN algorithms

— "**direct**" algorithm:
to join with key-value tables by direct lookups a.k.a. nested loops.
Good if the left table is small, but the right table is like a large dictionary.
Good to use in MATERIALIZED VIEW queries.

— "**parallel_hash**" algorithm:
speed-up if the right hand table is large.

— "**full_sorting_merge**" algorithm:
when right hand side is large
and does not fit in memory and does not allow lookups.

— "**grace_hash**" algorithm:
since in 22.12.

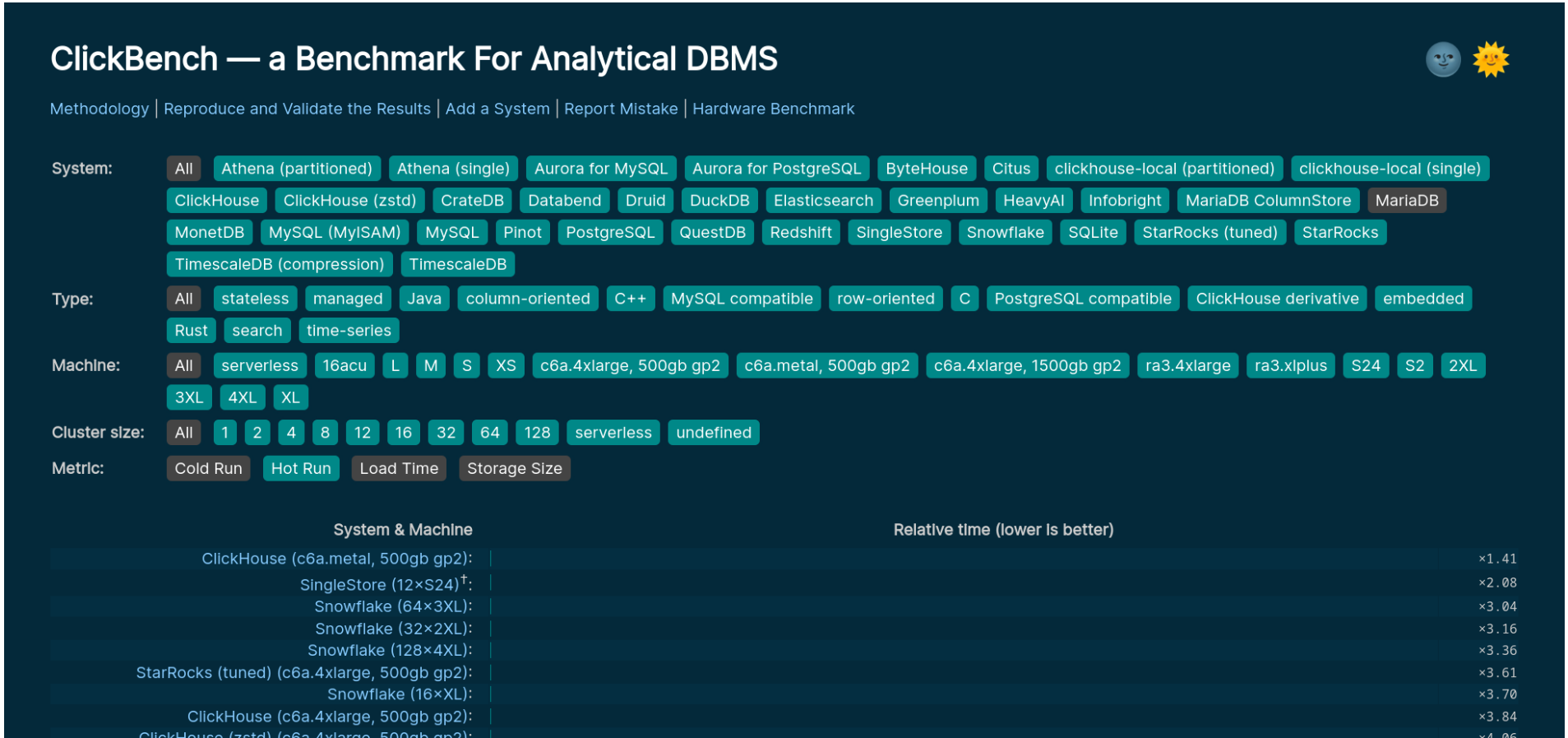
Developer: Vladimir Cherkasov, lgbo-ustc.

More Performance

... and we are reading from object storage 100 times faster.

Developer: Ksenia Sumarokova.

Updated Benchmark



Integrations

ClickHouse integrates with everything!

Integrations

ClickHouse can work **as a server** (clickhouse-server)
or **as a tool** without installation (clickhouse-local).

ClickHouse can **store the data**
or process externally stored data **on the fly**.

External data:
— remote databases: MySQL, PostgreSQL, MongoDB, ODBC, JDBC...
— object storages: S3, HDFS, Azure, COSN, OSS...
— from URL and local files;

All possible data formats:
— text: CSV, TSV, JSON, Values, MySQLDump, Regexp...
— binary: Parquet, Arrow, ORC, Avro, Protobuf, MsgPack...
— schemaful and schemaless;

Data Lakes

Now ClickHouse supports **Apache Hudi** and **Delta Lake**
for SELECT queries.

TODO: Apache Iceberg.

Advantages:
— these formats are somewhat resembling **MergeTree**
allowing incremental data insertion,
approaching to ClickHouse data formats;

Disadvantages:
— alien data formats from Apache/Hadoop/Java world;
— nothing works out of the box
unless you really know how to deal with it;

Data Lakes

Now ClickHouse supports **Apache Hudi** and **Delta Lake**
for SELECT queries.

```
SELECT count() FROM deltaLake(
  'https://clickhouse-public-datasets.s3.amazonaws.com/delta_lake/hits/')
  WHERE URL LIKE '%google%'

-- 4.396 sec.
```

Developers: Daniil Rubin, Ksenia Sumarokova, Flynn ucasfl. Since 22.11.

Integrations

— Querying MongoDB and Meilisearch with table functions.

— Streaming data consumption from NATS.

Developer: Anastasia Petrenko, Ksenia Sumarokova. Since 22.7.

Integrations

Visualizations:
— official ClickHouse plugin for **Grafana**;
— official support for **Superset**;
— **HEX** and **Deepnote** support.

Data ingestion and processing:
— **Kafka Connect** integration;
— **Airflow**, **dbt** support.

Language drivers:
— official **Node.JS** driver;
— optimized **Go** driver;
— a new **Python** client.

Operations

ClickHouse is easy to configure for your needs.

Self-Extracting Executable

The most simple way to install ClickHouse:

```
curl https://clickhouse.com/ | sh
```

Single binary package. Installs the latest version. Includes debug info.

Works on every **Linux** (x86_64, aarch64, powerpc64le),
macOS (x86_64, M1), **FreeBSD** and **Windows** (WSL2).

Was: **2.1 GB**.

Now: **446 MB**, takes ~5 seconds to decompress on first run.

Developer: Arthur Filatenkov, Yakov Olkhovskiy.

Composable Protocols

So, ClickHouse supports a lot of protocols:

— HTTP
— HTTPs
— Native TCP
— Native TCP wrapped in PROXYv1
— Native TCP with TLS
— MySQL (with TLS support)
— PostgreSQL (with TLS support)
— GRPC (with TLS)
— Replication protocol over HTTP
— Replication protocol over HTTPs
— Keeper client-server protocol;
— Keeper consensus protocol;
— ...

Composable Protocols

So, ClickHouse supports a lot of protocols.

How to configure all of them? What if:

— server has multiple network interfaces?
— enable one protocol on multiple ports?
— I want native TCP for localhost only and HTTPs from everywhere?
— I want different TLS certificates for different protocols?
— I want to wrap one protocol in another?

```
<protocols>
  <tcp>
    <type>tcp</type>
    <host>::</host>
    <port>9000</port>
    <description>native protocol</description>
  </tcp>
  <tcp_secure>
    <type>tls</type>
    <impl>tcp</impl>
    <port>9440</port>
    <description>secure native protocol</description>
  </tcp_secure>
  <tcp_endpoint>
    <impl>tcp</impl>
    <host>0.0.0.0</host>
    <port>9001</port>
    <description>native protocol, another</description>
  </tcp_endpoint>
  <tcp_proxy>
    <type>proxy1</type>
    <impl>tcp</impl>
    <port>9100</port>
    <description>native protocol with PROXYv1</description>
  </tcp_proxy>
```

Composable Protocols

The case: ClickHouse under proxy:

Envoy Proxy / HAProxy / CloudFlare.

ClickHouse server will receive connections from the proxy.

But it needs to know the source IP address for quotas, ACL and logging.

Solution: enable PROXYv1 protocol in the proxy
and configure it as a protocol wrapper in ClickHouse.

ClickHouse will read the header and unwrap the network packets.

Developer: Yakov Olkhovskiy. Since 22.10.

Compliance

CCPA

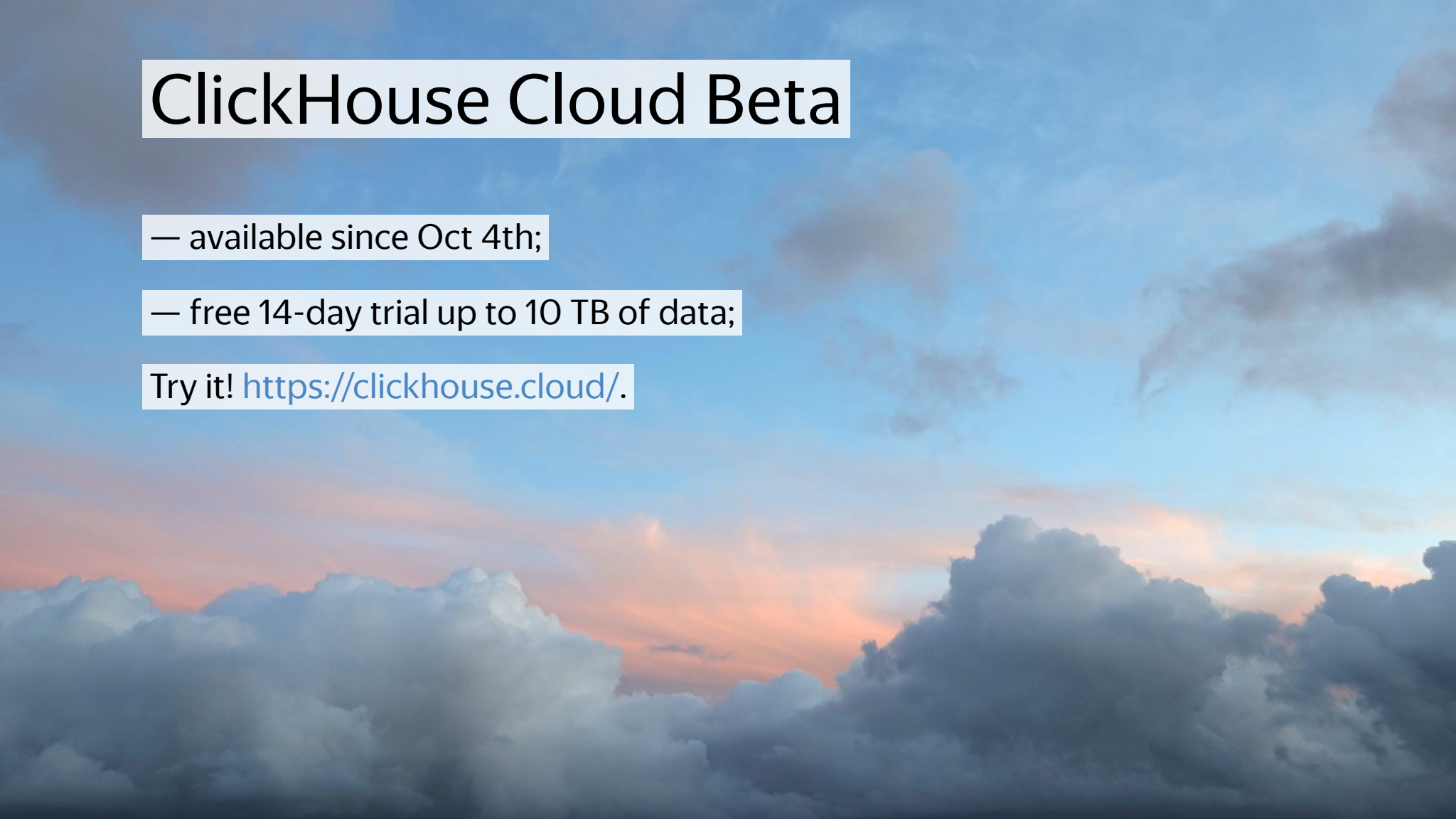
GDPR

PCI DSS

SOC 2

https://trust.clickhouse.com/

+ penetration testing, bug bounty program, audit reports...


ClickHouse Cloud Beta
— available since Oct 4th;
— free 14-day trial up to 10 TB of data;
Try it! https://clickhouse.cloud/.

Q&A