ClickHouse

Enabling interactive data exploration

Yandex

What is ClickHouse?

- RDBMS for analytics
  - SQL
  - FOSS
- Distributed
  - Cross-datacenter replication
  - Tolerant to a single-datacenter failure
- Linear scaling
  - 100s of servers
  - 100G-1000G records daily
- Blazing fast
  - interactive data exploration

These slides have a lot of links!

Download them at <insert link and code>

History

- Yandex.Metrica (think Google Analytics)
  - 30B rows daily, 3PB total, 700 machines
  - Processing speed up to 2TB/s
- Used a MyISAM solution with pre-aggregation
  - Didn’t scale for the growing load
  - Could only build pre-defined reports
- Started developing own columnar DB in 2009
- Enables “double real time” reporting
  - Add new events in real time
  - Build custom reports in real time
- Becomes the main engine in 2012
- Open-sourced in 2016

Adoption

- Yandex-wide
  - business analytics
- Thousands of companies worldwide
  - analyzing 1M DNS queries per second (Cloudflare)
  - geospatial processing (Carto)
  - real-time ad analytics platform (LifeStreet)
  - storage performance analysis and monitoring (Infinidat)
  - AdTech, FinTech, sensors, logs, event streams, time series, etc.

Unusual applications:

- Blockchain transaction history
  - blockchair.com
  - bloxy.info
- CERN LHCb experiment
- Bioinformatics

When to use

- stream of well-structured, immutable events
  - (mostly) fixed schema
  - append-only
  - heavy-weight ALTER UPDATE/DELETE as a “GDPR escape hatch”
- flexible real-time reporting
  - queries finish in seconds, not hours
  - no preprocessing needed
  - enables ad-hoc experiments

When NOT to use

- OLTP
  - no transactions
  - single INSERT is atomic, but no cross-query atomicity
  - very heavy UPDATEs
- key/value storage
  - sparse indexes
  - not suitable for point reads
- document/blob storage
  - optimized for records < 100 kB
- highly normalized data
  - optimized for star schema

How fast?

More on performance at our site

1.1 Billion Taxi rides benchmark: 1.1G records, 51 columns, 500 GB uncompressed CSV

Query 1	Query 2	Query 3	Query 4	Setup
0.005	0.01	0.10	0.188	BrytlytDB 2.1 & 5-node IBM Minsky cluster
0.051	0.15	0.05	0.794	kdb+/q & 4 Intel Xeon Phi 7210 CPUs
0.241	0.83	1.21	1.781	ClickHouse, 3 x c5d.9xlarge cluster
0.762	2.47	4.13	6.041	BrytlytDB 1.0 & 2-node p2.16xlarge cluster
1.034	3.06	5.35	12.748	ClickHouse, Intel Core i5 4670K
1.56	1.25	2.25	2.97	Redshift, 6-node ds2.8xlarge cluster
2	2.00	1.00	3	BigQuery
2.362	3.56	4.02	20.412	Spark 2.4 & 21 x m3.xlarge HDFS cluster
14.389	32.15	33.45	67.312	Vertica, Intel Core i5 4670K

Why so fast?

Read less data

- Data locality
  - columnar storage
  - sorted by PK
- Compression

Process data faster

- Parallelism
  - multithreading
  - distributed queries
- Efficient computation
  - 17 different GROUP BY algorithms
  - vectorized query execution with SIMD

Deployment

- Repos for major Linux distros
- Docker containers
- One self-contained binary
  - works everywhere
- ZooKeeper if you need replication
- Test it on your laptop
  - runs with minimal resources

- Stable release every two weeks
  - Thousands of scenarios tested for each change
- No data migration on update
  - Just run a new server

Data ingestion

- INSERTs
- Many data formats
  - CSV, JSON, Parquet, CapnProto, ORC, ...

- Batching for optimal performance
  - Buffer table engine
  - Kafka table engine
  - Third-party solutions — chproxy, kittenhouse etc.

Analyzing data

- A rich SQL dialect
  - strong typing
  - higher order functions like *arrayMap*
  - variety of aggregate function — quantiles, cardinality estimators etc.
  - sampling
  - *Nested* type for key/value records
  - *LowCardinality* type for dictionary encoding
- BI tools support
  - Holistics (Singapore-based)
  - Tableau
  - Apache Superset
  - others via ODBC/SQLAlchemy/...

clickhouse-local

The full power of ClickHouse engine over a data file.

```
$ clickhouse local
  --file ~/hits_v1.tsv
  --structure 'WatchID UInt64,  JavaEnable UInt8, ...'
  --query 'SELECT UserID, SearchPhrase, count() FROM table GROUP BY UserID,
SearchPhrase'

Read 8873898 rows, 7.88 GiB in 5.208 sec., 1704038 rows/sec.,  1.51 GiB/sec.
             UserID   SearchPhrase                 count()
8410854169855355129   пальные кость играть терхи         3
```

Interfacing with other systems

Connect to ClickHouse

- Native binary protocol
  - Drivers for Python, Go, C++, ...
- RESTful HTTP
- ODBC
- JDBC
- MySQL wire protocol
- PostgreSQL
  - clickhouse_fdw
  - pg2ch (logical replication)

Connect from ClickHouse

- File
- HDFS
- URL
- MySQL
- ODBC
- External dictionaries

Getting help

- Check the docs at our site
- Create issues on github
- Ask on Stack Overflow
- Email us at clickhouse-feeback@yandex-team.ru
- Join English and Russian chats in Telegram
- Get commercial support from Altinity and others
- … and more

Thank you!

Questions?

- https://clickhouse.yandex
- https://github.com/ClickHouse/ClickHouse
- clickhouse-feedback@yandex-team.ru

Yandex