ClickHouse – Open Source OLAP DBMS

introduction & cibersecurity applications

João Pinheiro - 2022

BLACKSHIELD

{ Overview }

BLACKSHIELD

{ What is Clickhouse }

- Open Source OLAP database, created by Yandex;

- True column-oriented approach (no extra data stored with values);

- Optimized data compression;

- Designed to work with regular hard drives;

- Distributed query support via sharding;

- Suitable for online queries – sub-second latencies;

- Data replication (async multi-master);

- Queriable via SQL Query language;

BLACKSHIELD

{ Typical usage scenarios }

- Small to large datasets with billions or trillions of rows;

- Data is organized in “wide” tables, comprised of many columns;

- The result dataset of a given query is a limited set of rows;

- Common query operations are aggregation queries;

- Results must be obtained in seconds or less;

BLACKSHIELD

{ Features }

- ClickHouse is fast. And by fast, I mean FAST.

- Easy to deploy & maintain, including scaling horizontally and backup;

- Multiple ingestion sources (SQL, CSV/TSV, S3, Kafka, etc);

- Supports foreign tables (MySQL, MongoDB, PostgreSQL, etc);

- Rich data types, suitable for many different usage scenarios;

- Multiple table engines suitable for different applications;

- Easy to use (if you know SQL);

- Integrates with existing dashboarding tools like grafana;

- Did I mention how fast it is?

BLACKSHIELD

{ Pitfalls & limitations }

- No “proper” transactions;

- Datasets are mostly “insert-read”;

- Limited functionality on updates & deletes and they are processed as batch operations;

- Limited performance on point queries returning a single row by key;

- No table relations;

- Works better with denormalized data;

- Inserts should be batched – it takes roughly the same time to insert 10 rows or 1000 rows;

BLACKSHIELD

{ ClickHouse: data types }

Integer: Uint{8,16,32,64,128,256}, Int{8,16,32,64,128,256}

Variable precision: Float32, Float64

Fixed precision: Decimal

String/Char/BLOB: String, FixedString(n)

Date/Time: Date, Date32, DateTime, DateTime64

Geo: Point, Ring, Polygon, MultiPolygon

Misc: Boolean, UUID, LowCardinality, Enum, Array, JSON*, Nested

Network: Ipv4, IPv6

BLACKSHIELD

Learn More: https://clickhouse.com/docs/en/sql-reference/data-types/

{ ClickHouse: table engines }

*MergeTree engine

- Quick inserts with background merging;
- Data replication with Replicated* engines;
- Optional partitioning via partitioning key;
- Supports primary key/sorting key;
- Advanced features: sampling & TTL;

* Log engine

- Small tables, up to 1M records;
- Supports HDFS and S3 as file systems;
- Does not support indexes nor mutations;

Memory engine

- Uses RAM without compression;
- No overhead on read operations;
- Does not support indexes;
- Non-persistent;

Integration engines

- ODBC/JDBC;
- MySQL, MongoDB,PostgreSQL, RocksDB;
- HDFS, S3;
- Kafka, RabbitMQ, File;

Learn More: https://clickhouse.com/docs/en/engines/table-engines/

{ showcase }

importing a dataset for quick analysis

BLACKSHIELD

Ingesting data from a well-known leak: install ClickHouse

{ Showcase }

```
sudo apt-get install -y apt-transport-https ca-certificates dirmngr
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
8919F6BD2B48D754

echo "deb https://packages.clickhouse.com/deb stable main" | sudo tee \
    /etc/apt/sources.list.d/clickhouse.list
sudo apt-get update

sudo apt-get install -y clickhouse-server clickhouse-client

sudo service clickhouse-server start
clickhouse-client # or "clickhouse-client --password" if you've set up a
password.
```

Machine:

Hetzner Cloud VPS
Ubuntu 22.04
3x AMD Epyc 2.5Ghz
4GB RAM
80GB SSD

Source to be ingested:

19GB of CSV files

BLACKSHIELD

Learn More: https://clickhouse.com/docs/en/install#self-managed-install

Ingesting data from a well-known leak: create a table

{ Showcase }

```
CREATE TABLE IF NOT EXISTS tap(
    MDM_ID String,
    BIRTH_DATE String,
    SALUTATION String,
    COMMUNICATION_LANGUAGE LowCardinality(String),
    FIRST_NAME String,
    LAST_NAME String,
    FULL_NAME String,
    NATIONALITY LowCardinality(String),
    GENDER LowCardinality(String),
    ADDRESS_CITY LowCardinality(String),
    ADDRESS_COUNTRY LowCardinality(String),
    ADDRESS_DETAIL String,
    ADDRESS_REGION String,
    ADDRESS_ZIPCODE String,

(...)

    STATUS LowCardinality(String),

    INDEX idx01 (ADDRESS_CITY) TYPE set(0) GRANULARITY 1

) ENGINE = MergeTree()
    ORDER BY (ADDRESS_COUNTRY, FIRST_NAME, LAST_NAME);
```

Machine:

Hetzner Cloud VPS
Ubuntu 22.04
3x AMD Epyc 2.5Ghz
4GB RAM
80GB SSD

Source to be ingested:

19GB of CSV files

BLACKSHIELD

Learn More: https://clickhouse.com/docs/en/sql-reference/

Ingesting data from a well-known leak: ingest data

{ Showcase }

Import bash script

```
for FILENAME in /data/src/Customer_*.csv; do
    echo $FILENAME
    clickhouse-client --query="INSERT INTO tap FORMAT CSV" \
      --format_csv_delimiter="|" < $FILENAME
done
```

```
root@ubuntu-4gb-fsn1-2:/data# time ./clickhouse-import.sh

(...)

real    12m8.895s
user    9m23.545s
sys     0m34.991s
root@ubuntu-4gb-fsn1-2:/data#
```

Give it 2-3 Minutes to settle and check disk usage

```
root@ubuntu-4gb-fsn1-2:/data# du -h /var/lib/clickhouse/

(...)

3.9G    /var/lib/clickhouse/

root@ubuntu-4gb-fsn1-2:/data#
```

Machine:

Hetzner Cloud VPS
Ubuntu 22.04
3x AMD Epyc 2.5Ghz
4GB RAM
80GB SSD

Source to be ingested:

19GB of CSV files

ClickHouse disk usage:

3.**9GB**

BLACKSHIELD

Ingesting data from a well-known leak: query time!

{ Showcase }

How many records

```
ubuntu-4gb-fsn1-2 :) select count(*) from tap;

SELECT count(*)
FROM tap

Query id: 0f315a2e-2419-4254-b8d7-83a891480cd5

┌─count()─┐
│ 77629463 │
└──────────┘

1 row in set. Elapsed: 0.003 sec.
```

77 Million records in 12m8s:

~ 6.4M inserts/minute
~10600 inserts/second

How many unique e-mail addresses

```
ubuntu-4gb-fsn1-2 :) select count(distinct(EMAIL_DESCRIPTION)) from tap;

SELECT countDistinct(EMAIL_DESCRIPTION)
FROM tap

Query id: 02f3174e-1e29-470d-b52e-7546ad952b41

┌─uniqExact(EMAIL_DESCRIPTION)─┐
│                      6075465 │
└──────────────────────────────┘

1 row in set. Elapsed: 4.170 sec. Processed 77.63 million rows, 2.48 GB
(18.62 million rows/s., 593.82 MB/s.)
```

BLACKSHIELD

Ingesting data from a well-known leak: query time!

{ Showcase }

How many records for someone with ‘Manuel’ on the name and from Gouveia city (case sensitive)

```
ubuntu-4gb-fsn1-2 :) select count(*) from tap where ADDRESS_CITY='Gouveia' and FULL_NAME like '%Manuel %';

SELECT count(*)
FROM tap
WHERE (ADDRESS_CITY = 'Gouveia') AND (FULL_NAME LIKE '%Manuel %')

Query id: a49fb40d-b44f-4120-9ef2-9894420ccabe

┌─count()─┐
│      45 │
└─────────┘

1 row in set. Elapsed: 0.417 sec. Processed 6.29 million rows, 144.64 MB (15.09 million rows/s., 346.86 MB/s.)
```

BLACKSHIELD

Ingesting data from a well-known leak: query time!

{ Showcase }

Top10 of records per Country

```
ubuntu-4gb-fsn1-2 :) select ADDRESS_COUNTRY,count(*) as total from tap group by(ADDRESS_COUNTRY) order by total
desc limit 10;

(...)

┌─ADDRESS_COUNTRY─┬────total─┐
│ BR              │ 21682463 │
│ PT              │ 17130056 │
│ US              │  5911385 │
│ FR              │  4952701 │
│ GB              │  4057600 │
│ ES              │  3555442 │
│ DE              │  3146254 │
│                 │  2603944 │
│ IT              │  2578419 │
│ CH              │  1961251 │
└─────────────────┴──────────┘

10 rows in set. Elapsed: 0.210 sec. Processed 77.63 million rows, 81.77 MB (369.45 million rows/s., 389.14 MB/s.)
```

BLACKSHIELD

Ingesting data from a well-known leak: export data

{ Showcase }

Export all e-mail addresses to a CSV file

```
ubuntu-4gb-fsn1-2 :) select distinct(EMAIL_DESCRIPTION) from tap into outfile 'emails.csv' format CSV;

SELECT DISTINCT EMAIL_DESCRIPTION
FROM tap
INTO OUTFILE 'emails.csv'
FORMAT CSV

Query id: 245fa26a-b302-4456-8570-194c52885974


6075465 rows in set. Elapsed: 5.542 sec. Processed 77.63 million rows, 2.48 GB (14.01 million rows/s., 446.80
MB/s.)

ubuntu-4gb-fsn1-2 :)
```

BLACKSHIELD

{ showcase }

querying network events

BLACKSHIELD

Using ClickHouse to query network-related events

{ showcase }

Event table

```
 CREATE TABLE IF NOT EXISTS Events
(
    Id UUID,
    Created DateTime32,
    Address IPv4,
    Domain String
) ENGINE = MergeTree()
    ORDER BY (Id, Created)
    PRIMARY KEY Id;
```

Machine:

Hetzner Cloud VPS
Ubuntu 22.04
3x AMD Epyc 2.5Ghz
4GB RAM
80GB SSD

Dataset Size:
149.997.000 Events

Sample data

```
┌─Id───────────────────────────────────┬─────────────Created─┬─Address────────┬─Domain────────────┐
│ 911008ec-7681-46a2-8000-066509e1f838 │ 2022-03-04 10:20:40 │ 76.172.30.184  │ ggpewp.vhn        │
│ f4804030-56b2-4776-8000-1a67631308b2 │ 2021-08-13 22:51:54 │ 146.193.180.94 │ ykkxgckrtscs.vbr  │
│ 1889327e-8f14-4b61-8000-20da5fef938e │ 2021-07-25 09:03:05 │ 195.33.251.217 │ vxscmq.fim        │
│ 7ff7fddf-4fce-47b3-8000-4a7330474c55 │ 2021-02-11 18:42:10 │ 115.174.75.31  │ kdkrgkgwna.eno    │
│ eeb954eb-8c2b-4187-8000-55b3c318b599 │ 2022-08-06 12:40:44 │ 217.63.148.2   │ zuulwdjpqkxn.aqr  │
│ bb6ea2f1-cf55-46a5-8000-832dc58c428f │ 2021-01-13 23:46:01 │ 7.226.57.213   │ mouboknjhukz.loy  │
```

BLACKSHIELD

Using ClickHouse to query network-related events

{ showcase }

List first 10 records whose IP is within the subnet 10.0.0.0/16

```
ubuntu-4gb-fsn1-2 :) SELECT Created, Domain, Address FROM Events
WHERE isIPAddressInRange(IPv4NumToString(Address), '10.0.0.0/16')=1
LIMIT 10;

(...)

┌─────────────Created─┬─Domain─────────┬─Address─────┐
│ 2022-08-17 06:31:30 │ pcxjjtjhym.acr │ 10.0.218.42 │
└─────────────────────┴────────────────┴─────────────┘

10 rows in set. Elapsed: 0.041 sec. Processed 712.70 thousand rows, 4.00 MB (17.25 million rows/s., 96.83
MB/s.)
```

Learn More: https://clickhouse.com/docs/en/sql-reference/functions/ip-address-functions

BLACKSHIELD

Using ClickHouse to query network-related events

{ showcase }

Retrieve the first 10 records that match:
- Address matches both 10.0.0.0/8 and 10.0.10.0/16
- TLD of domain starts with letter ‘a’
- Address has a 2 somewhere in it
- Created between 01-01-2022 and 01-10-2022
- Older events first

```
ubuntu-4gb-fsn1-2 :) SELECT Created, Domain, IPv4NumToString(Address) AS addr
                     FROM Events
                     WHERE isIPAddressInRange(addr, '10.0.0.0/8')=1
                     AND Domain like '%.a%'
                     AND addr like '%2%'
                     AND isIPAddressInRange(addr, '10.0.10.0/16')=1
                     AND Created BETWEEN '2022-01-01 00:00:00' AND '2022-10-01 00:00:00'
                     ORDER BY Created
                     LIMIT 10;
```

BLACKSHIELD

Learn More: https://clickhouse.com/docs/en/sql-reference/functions/date-time-functions

Using ClickHouse to query network-related events

{ showcase }

```
(...)

┌─────────────Created─┬─Domain───────────┬─addr─────────┐
│ 2022-01-02 15:06:59 │ mhzebj.anh       │ 10.0.18.244  │
│ 2022-02-20 02:24:42 │ apcquscimhbv.abu │ 10.0.48.239  │
│ 2022-03-02 13:39:15 │ qzlclgsljkv.aqz  │ 10.0.55.223  │
│ 2022-03-19 05:04:59 │ crcijzzjbdui.alo │ 10.0.140.243 │
│ 2022-03-27 20:09:38 │ fenyyesjiq.aol   │ 10.0.241.240 │
│ 2022-04-06 07:52:18 │ bsyfnrympr.amp   │ 10.0.122.175 │
│ 2022-04-06 21:30:37 │ dvskgbfnab.anq   │ 10.0.207.63  │
│ 2022-05-01 20:22:16 │ wplaxuv.arr      │ 10.0.228.97  │
│ 2022-05-05 18:32:27 │ bxmdouxwa.agd    │ 10.0.15.212  │
│ 2022-05-05 19:56:59 │ gqeqldkip.adg    │ 10.0.2.241   │
└─────────────────────┴──────────────────┴──────────────┘

10 rows in set. Elapsed: 7.897 sec. Processed 150.00 million rows, 4.50 GB (18.99 million rows/s., 569.80
MB/s.)
```

BLACKSHIELD

{ thank you }

questions?

BLACKSHIELD

{ Online Resources }

Project page: https://clickhouse.com/

Tutorials and datasets: https://clickhouse.com/docs/en/getting-started/example-datasets/

Playground with datasets: https://clickhouse.com/docs/en/getting-started/playground

Altinity Blog: https://altinity.com/blog/

Curated list of resources: https://github.com/korchasa/awesome-clickhouse

Interactive Benchmark: https://benchmark.clickhouse.com/

BLACKSHIELD