Yandex
ClickHouse

High-Performance Distributed DBMS for Analytics
About me

› Developer, hardware engineering background

› Head of Analytic Products Department in Yandex

› jkee@yandex-team.ru
About Yandex

› One of the largest internet companies in Europe

› Over 5000 employees

› Top-1 Search in Russia

› More than 50 different b2c and b2b products

› Big Data, Machine Learning
ClickHouse

History
The Product

Yandex.Metrica is one of the largest web analytic tools

- 20+ billions of events daily
- Millions of websites
- 100+ thousands of analysts every day

We need fast and feature-rich database capable to handle our clients data
Yandex.Metrica

Traffic source

Sessions

2016.09.19 10:30, Monday
Internal traffic
Sessions: 374

- Direct traffic: 49.3%
- Link traffic: 24.5%
- Internal traffic: 18.3%
- Search engine traffic: 7.67%
- Social network traffic: 0.099%
- Other: 0.038%
Aggregated or not

"Classic" approach:
- aggregated data
- a lot of tables

Modern tool:
- various filters, views
- unaggregated data
- few tables

Key -> Metrics
Event -> Properties
We really need something different

Our requirements was:

› Fast. Really fast
› Data processing in real time
› Capable of storing petabytes of data
› Fault-tolerance in terms of datacenters
› Flexible query language
Oh. Nothing? Well...
The main ideas behind ClickHouse

› SQL dialect + extensions
› Linearly scalable
› Focused on fast query execution
› Column-oriented
ClickHouse timeline

- **Prototype**: Jan 2009
- **Production**: Aug 2012
- **Data Transfer**: June 2014
- **Metrica 2.0**: 9 Dec 2014
- **Open Source**: 15 June 2016
ClickHouse today

› More than 20 projects in Yandex
› Open-source from June 2016
› Production outside Yandex
ClickHouse

Features
Linearly scalable

› Petabytes of data
› Cross-datacenter
› High availability
› Data compression
Metrica Cluster

› 3 Pb
› 412 Nodes
› 6 Datacenters
› Less than few hours of downtime in 4 years
Querying

› SQL dialect + extensions
› Additional features: approximate functions, URI functions and more
› Arrays, nested data types
› All distributed (including JOIN)
› Pluggable external key-value sources
# Weekly traffic and audience

SELECT
  count() as visits,
  sum(PageViews) as hits,
  uniq(UserID) as users
FROM visits_all
WHERE StartDate > today() - 7
# Using external dictionary for regions

```sql
SELECT
count() as visits,
regionToName(regionToCountry(RegionID), 'en') as country
FROM visits_all
WHERE StartDate > today() - 7
GROUP BY country
ORDER BY visits DESC
LIMIT 10
```
So fast it makes you think faster

› Sub-second query latency
› >100x faster than Hadoop,
  >100x faster than typical DBMS
› 100,000 - 1,000,000 rows/second
  on a single node
› Up to 2 terabytes per second
  clustered setup of 400 nodes
Without ClickHouse
Make a query and go for a coffee
Or explain to manager why it's so slow

With ClickHouse
Check 37 ideas in 5 minutes
WARNING: no way back
Relative query processing time (lower is better):

<table>
<thead>
<tr>
<th>Database</th>
<th>Version</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClickHouse</td>
<td>1.1.53960</td>
<td>1.00</td>
</tr>
<tr>
<td>Hive</td>
<td>0.11, ORC File</td>
<td>168.19</td>
</tr>
<tr>
<td>MySQL</td>
<td>5.5.32, MyISAM</td>
<td>511.57</td>
</tr>
</tbody>
</table>

Relative query processing time (lower is better):

<table>
<thead>
<tr>
<th>Database</th>
<th>Version</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClickHouse</td>
<td>1.1.53960</td>
<td>1.00</td>
</tr>
<tr>
<td>Vertica</td>
<td>7.1.1</td>
<td>3.38</td>
</tr>
<tr>
<td>InfiniDB</td>
<td>Enterprise 3.6.23</td>
<td>20.91</td>
</tr>
<tr>
<td>MonetDB</td>
<td></td>
<td>21.51</td>
</tr>
</tbody>
</table>

More info: https://clickhouse.yandex/benchmark.html
Interfaces

› Console client
› HTTP
› JDBC Driver
› Python, PHP, NodeJS, Go and Perl wrappers
Why it's so fast?

- Code
  - Vectorized query execution
  - Performance-oriented well-written C++14 code
  - Every piece of code is tested in terms of performance
Why it's so fast?

- Data
  - Column-oriented
  - Merge Tree
    - minimal number of seeks
  - All processing as close to data as possible
Why it's so fast?

Features

› Sampling

› Approximate functions

› Performance tuning
  even on a request level
Scalability and fault-tolerance

- Cluster of shards, each shard is a group of replicas
- Asynchronous replication, eventual consistency
- ZooKeeper for replicas coordination (not used on select's)
Cluster Scheme

Data Center

Shard
Use cases
ClickHouse: wrong cases

› Not an OLTP
› Not a key-value store
› Not a document store
› Do not modify your data (you don't need that)
ClickHouse: best practices

› A few wide tables with a lot of columns
› QPS is relatively low but data usage per request is high
› Huge amounts of data incoming
› Petabytes of data
Case: server log analysis

Common first case for new ClickHouse users.
Estimated time: few hours

› Insert access logs into ClickHouse
› Analyze incidents with instant queries
› Monitoring reports: error rates, response timings and more
Case: in-house analytics database

Build your own data warehouse and dig your data in seconds.

› Take your Hadoop or other 'not so fast' storage and want to do things faster
› Copy all your data to ClickHouse
› Build internal dashboards/metrics
› Do realtime analysis of your business process
How to start?
Try our tutorial: https://clickhouse.yandex/tutorial.html

Feel free to ask anything: clickhouse-feedback@yandex-team.com

GitHub: https://github.com/yandex/ClickHouse

More info: https://clickhouse.yandex
Wrap up
ClickHouse briefly

- Open-source column-oriented DBMS
- Linearly scalable
- Blazingly fast
- SQL dialect with extensions
Rate My Session!

**Schedule**
Timezone: Europe/Berlin +02:00

**Details**
Introducing gh-ost: triggerless, painless, trusted online schema migrations

**Rate & Review**
Tap a star to rate

Feedback (optional)
Contacts

Viktor Tarnavskiy

Head of Analytic Products Department in Yandex

jkee@yandex-team.ru

clickhouse-feedback@yandex-team.ru