12th ClickHouse Meetup

in Beijing

Evolution of Data Structures

in Yandex.Metrica

About me

Alexey, developer of ClickHouse.

I work on data processing engine of Yandex.Metrica since 2008.

About Yandex

Yandex is one of the largest internet companies in Europe
operating Russia’s most popular search engine.

About Yandex

In short, Yandex is russian Baidu.

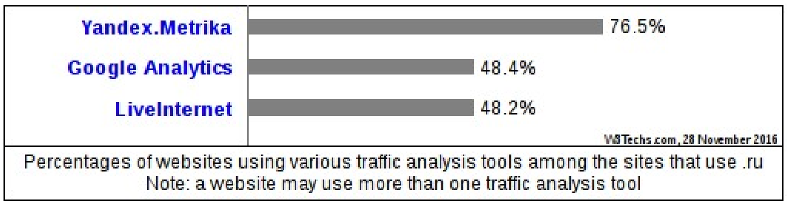
... or russian Sina

... or russian Google

About Yandex.Metrica

Yandex.Metrica (https://metrica.yandex.com/) is a service for web analytics.

Largest in Russia, second largest in the world (just after Google Analytics).

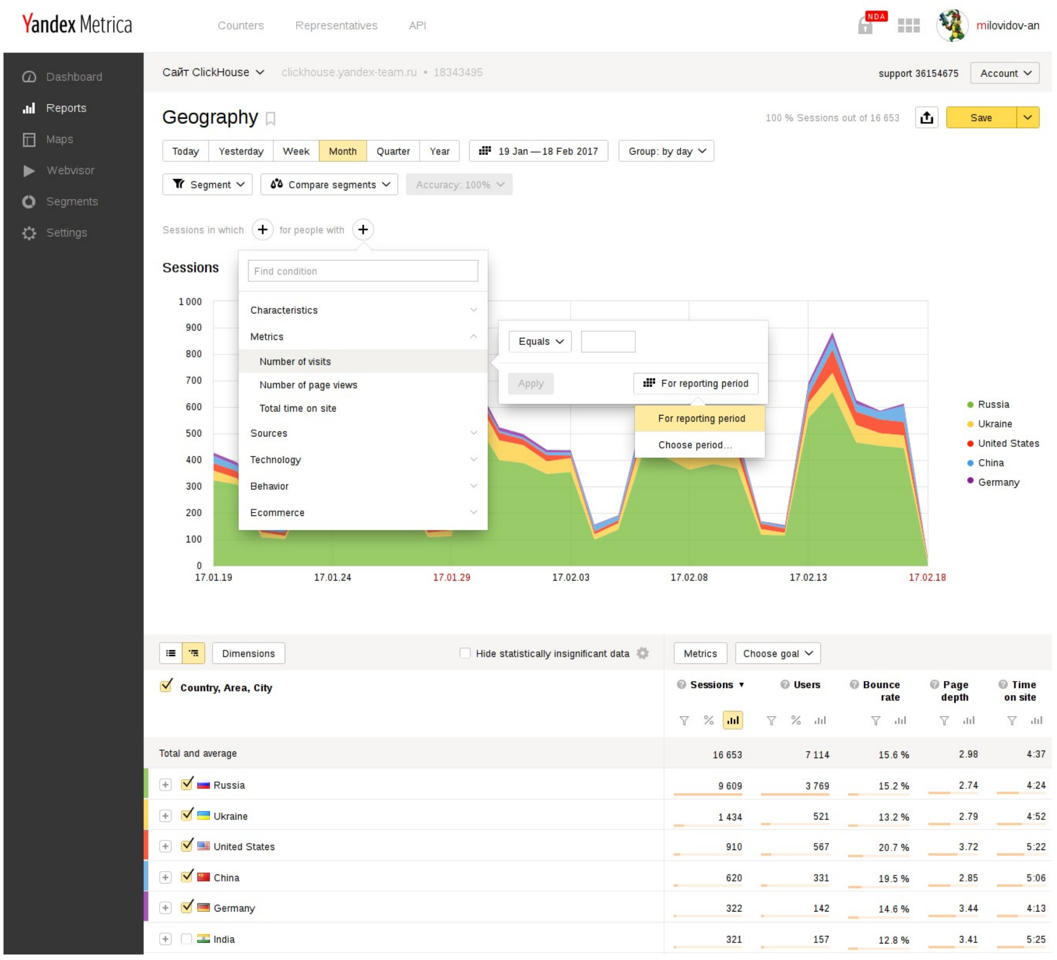


We are processing about ~25 billions of events (page views, conversions, etc).

We must generate and show reports in realtime.

About Yandex.Metrica

In short, Yandex.Metrica is a something like Baidu.Analytics



✓ Country, Area, City	Sessions ▾	Users	Bounce rate	Page depth	Time on site
Total and average	16 653	7 114	15.6 %	2.98	4:37
✓ Russia	9 609	3 769	15.2 %	2.74	4:24
✓ Ukraine	1 434	521	13.2 %	2.79	4:52
✓ United States	910	567	20.7 %	3.72	5:22
✓ China	620	331	19.5 %	2.85	5:06
✓ Germany	322	142	14.6 %	3.44	4:13
India	321	157	12.8 %	3.41	5:25

History

How to store data?

Big data processing is not a problem.

The challenge is how to store data in that way to allow both:

- efficient ingestion of click stream in realtime;

- efficient generation of reports;

Let review our historical solutions first...

MySQL (MyISAM) 2008-2011

We have had about 50 predefined report types.

We create a table for each of them.

Each table has primary key in form of:

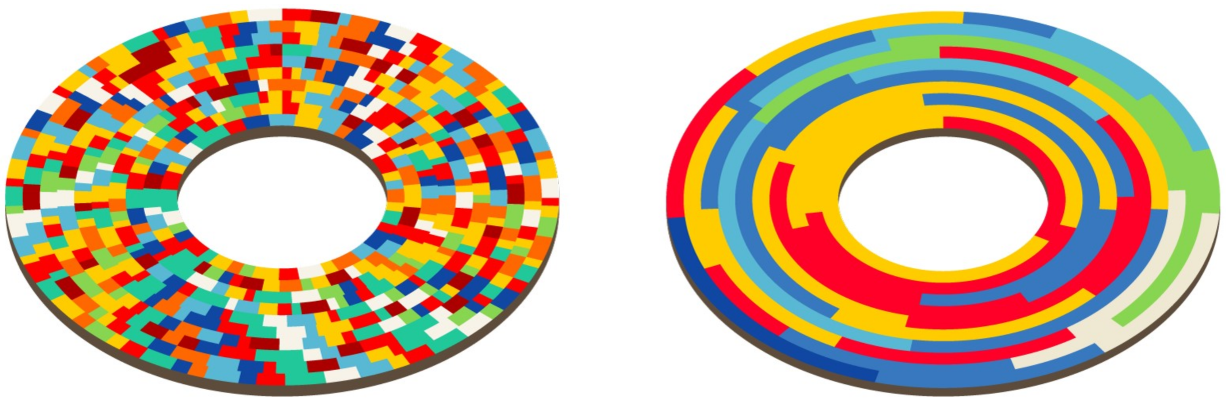
site_id, date, *key* -> aggregated statistics.

The data was inserted in mini-batches of aggregated deltas,
using ON DUPLICATE KEY UPDATE.

... but this just don't work.

Data locality on disk (artistic view)

The main concern is data locality.



MySQL (MyISAM) 2008-2011

We use HDD (rotational drives).
We cannot afford petabytes of SSDs.

Each seek is ~12 ms of latency,
usually no more than 1000 random reads/second in RAID array.

Time to read data from disk array is dependent on:
- number of seeks;
- total amount of data;

Example: read 100 000 rows, randomly scattered on disk:
- at least 100 seconds in worst case.
User won't wait hundred seconds for the report.

The only way to read data from disk array in appropriate amount of time is to minimize number of seek by maintaining data locality.

MySQL (MyISAM) 2008-2011

Fundamental problem:

Data is **inserted** almost in time order:
- each second we have new portion data for this second;
- but data for different web sites are comes in random order in a stream;

Data is **selected** by ranges for specified web site and date period:
- in ranges of completely different order;

MySQL (MyISAM) 2008-2011

MyISAM stores data in MYD and MYI files.
MYD contains data almost in order of insertion.
MYI contains B-tree index that maps a key to offset in MYD file.

Insertion of data is almost fine.
But selecting of data by range of primary key was non-practical.

Nevertheless, we made it work by:

- tricky partitioning;
- organizing data in few generations with different partitioning scheme;
- moving data between tables by scheduled scripts;
- report generation becomes ugly UNION ALL queries.

MySQL (MyISAM) 2008-2011

As of 2011 we was storing about 580 billion rows in MyISAM tables.

We were not satisfied by performance and maintenance cost:
Example: page title report loading time, 90% quantile was more than 10 seconds.

... After all, everything was converted and deleted.

Metrage, 2010-2015

(Specialized data structure, developed specially for aggregation of data and report generation).

To maintain data locality, we need
to constantly reordering data by primary key.

We cannot maintain desired order at INSERT time, nor on SELECT time;
we must do it in background.

Obviously: we need an LSM-tree!

Metrage, 2010-2015

Metrage: Metrica + Aggregated statistics.

We have created custom data structure for that purpose.

In 2010, there was no LevelDB.
We just got some insights from article about TokuDB.

Metrage is designed for the purpose of realtime data aggregation:
- row in Metrage table is custom C++ struct with *update* and *merge* methods.

Example: a row in Metrage table could contain a HyperLogLog.

Data in Metrage is aggregated:
- on insertion, in batches;
- during background compaction;
- on the fly, during report generation.

Metrage, 2010-2015

Everything was working fine.
The problem of data locality was solved.
Reports was loading quickly.

As of 2015 we stored 3.37 trillion rows in Metrage
and used 39 * 2 servers for this.

But we have had just ~50 pre-defined reports.

No customization and drill down was possible.

The user wants to slice and dice every report by every dimension!

... and we have developed just another custom data structure.

The report builder, 2010

We had quickly made a prototype of so-called "report builder".

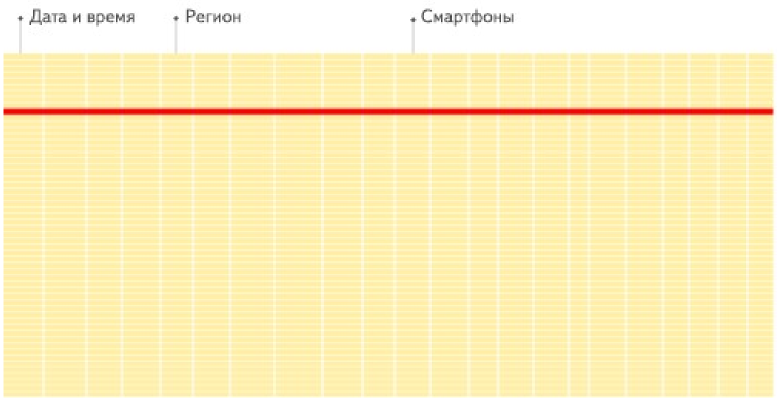
This was 2010 year. It was just simple specialized column-oriented data structure.

It worked fine and we got understanding, what the right direction to go.

We need good column-oriented DBMS.

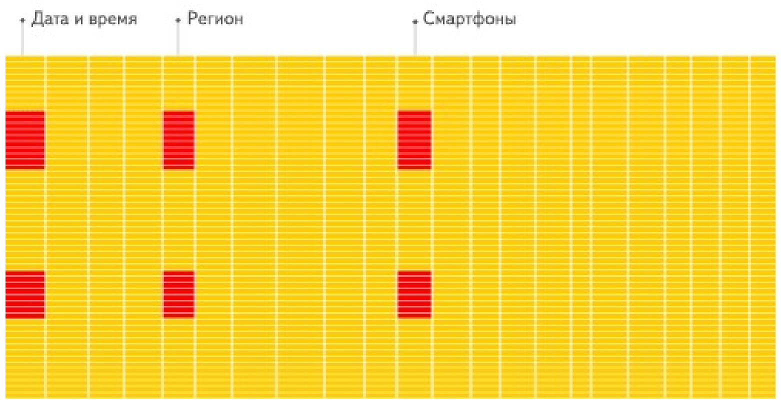
Why column-oriented?

This is how "traditional" row-oriented databases work:



Why column-oriented?

And this is how column-oriented databases work:



Why column-oriented?

Hypothesis:

If we have good enough column-oriented DBMS,
we could store all our data in non-aggregated form
(raw pageviews and sessions) and generate all the reports on the fly,
to allow infinite customization.

To check this hypothesis, we started to evaluate existing solutions.

MonetDB, InfiniDB, Infobright and so on...

No appropriate solutions were exist in 2010.

ClickHouse

As an experimental project, we started to develop
our own column-oriented DBMS: ClickHouse.

In 2012 it was in production state.

In 2014 we re-lauched Yandex.Metrica as Metrica 2.

All data is stored in ClickHouse and in non-aggregated form
and every report is generated on the fly.

In Metrika 2 the user could create it's own report with
- custom dimensions, metrics, filters, user-centric segmentation...
- and to dig through data to the detail of individual visitors.

ClickHouse

The main target for ClickHouse is query execution speed.

In Yandex.Metrika, users could analyze data for their web sites of any volume.

Biggest classifieds and e-commerce sites with hundreds millions PV/day are using Yandex.Metrika (e.g. ru.aliexpress.com).

In contrast to GA*, in Yandex.Metrika, you could get data reports for large web sites without sampling.

As data is processed on the fly, ClickHouse must be able to crunch all that pageviews in sub second time.

* in Google Analytics you could get reports without sampling only in "premium" version.

The main cluster of Yandex.Metrica

- 25 trillions of rows (as of Sep 2017)
- 500 servers
- total throughput of query processing is up to two terabytes per second

* If you want to try ClickHouse, one server or VM is enough.

ClickHouse

- column-oriented
- distributed
- linearly scalable
- fault-tolerant
- data ingestion in realtime
- realtime (sub-second) queries
- support of SQL dialect + extensions
  (arrays, nested data types, domain-specific functions, approximate query execution)

Open-source (since June 2016)

We think ClickHouse is too good to be used solely by Yandex.

We made it open-source. License: Apache 2.0.

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

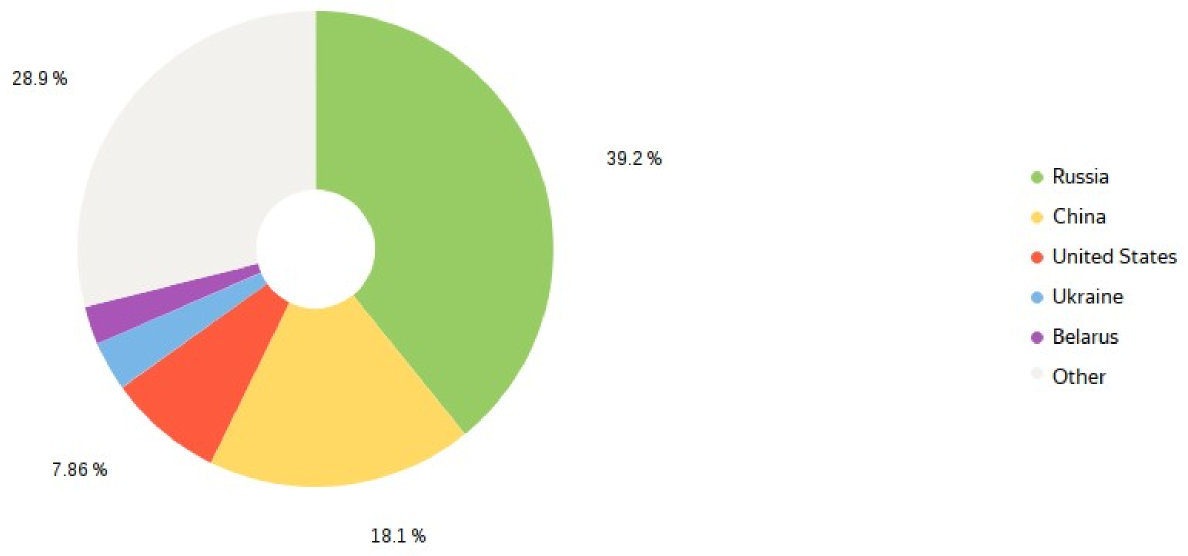
彪悍开源的分析数据库-ClickHouse

Open-source

More than 100 companies are already using ClickHouse.

Examples: Mail.ru, Cloudflare, Vertamedia, Lifestreet, Kaspersky...

Countries Breakdown



When to use ClickHouse

For well structured, clean, immutable events.

Click stream. Web analytics. Adv. networks. RTB. E-commerce.

Analytics for online games. Sensor and monitoring data. Telecom data.

When not to use ClickHouse

OLTP
ClickHouse doesn't have UPDATE statement and full-featured transactions.

Key-Value
If you want high load of small single-row queries, please use another system.

Blob-store, document oriented
ClickHouse is intended for vast amount of fine-grained data.

Over-normalized data
Better to make up single wide fact table with pre-joined dimensions.

Benchmarks

Evidencies...

Our CEO first saw ClickHouse in action:

geoff [1:54 PM]

I have a boner looking at the new stats
it's so fucking fast

Source: https://t.me/clickhouse_ru

Why ClickHouse is so fast?

— we just cannot make it slower.

Yandex.Metrica must work.

Why ClickHouse is so fast?

Algorithmic optimizations.

MergeTree, locality of data on disk
— fast range queries.

Example: uniqCombined function is a combination of three different data structures, used for different ranges of cardinalities.

Low-level optimizations.

Example: vectorized query execution.

Specialization and attention to detail.

Example: we have 17 different algorithms for GROUP BY. Best one is selected for your query.

Interfaces

HTTP REST

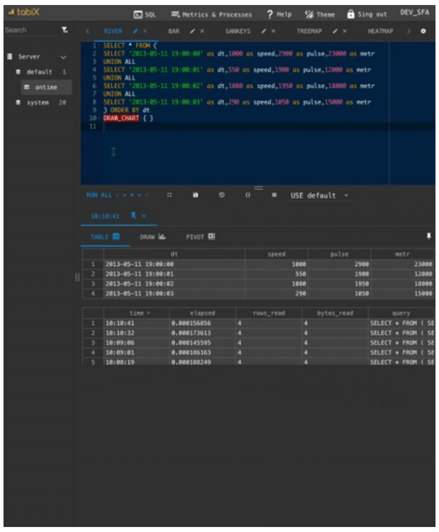
clickhouse-client

JDBC (production), ODBC (beta)

Python, PHP, Perl, Go,
Node.js, Ruby, C++, .NET, Scala, R, Julia, Rust

Integration with data visualization tools

Tabix (tabix.io) — developed specially for ClickHouse.



```
SELECT * FROM (
SELECT '2013-05-11 19:00:00' as dt,1000 as speed,2900 as pulse,23000 as metr
UNION ALL
SELECT '2013-05-11 19:00:01' as dt,550 as speed,1900 as pulse,12000 as metr
UNION ALL
SELECT '2013-05-11 19:00:02' as dt,1080 as speed,1950 as pulse,18000 as metr
UNION ALL
SELECT '2013-05-11 19:00:03' as dt,290 as speed,1050 as pulse,15000 as metr
) ORDER BY dt
DRAW_CHART { }
```

	dt	speed	pulse	metr
1	2013-05-11 19:00:00	1000	2900	23000
2	2013-05-11 19:00:01	550	1900	12000
3	2013-05-11 19:00:02	1080	1950	18000
4	2013-05-11 19:00:03	290	1050	15000

	time	elapsed	rows_read	bytes_read	query
1	10:10:41	0.000156856	4	4	SELECT * FROM (SE
2	10:10:32	0.000173613	4	4	SELECT * FROM (SE
3	10:09:06	0.000145595	4	4	SELECT * FROM (SE
4	10:09:01	0.000186163	4	4	SELECT * FROM (SE
5	10:08:19	0.000188249	4	4	SELECT * FROM (SE

And:

Grafana, Redash, Apache Zeppelin, Superset...
Tableau Desktop (basically works on Windows).

Community

Web site: https://clickhouse.yandex/

Google groups: https://groups.google.com/forum/#!forum/clickhouse

Maillist: clickhouse-feedback@yandex-team.com

Telegram chat (hope it isn't banned in China):
https://telegram.me/clickhouse_en and https://telegram.me/clickhouse_ru
(now with over 1300 members)

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

Twitter: https://twitter.com/ClickHouseDB

China User Group: http://clickhouse.com.cn/

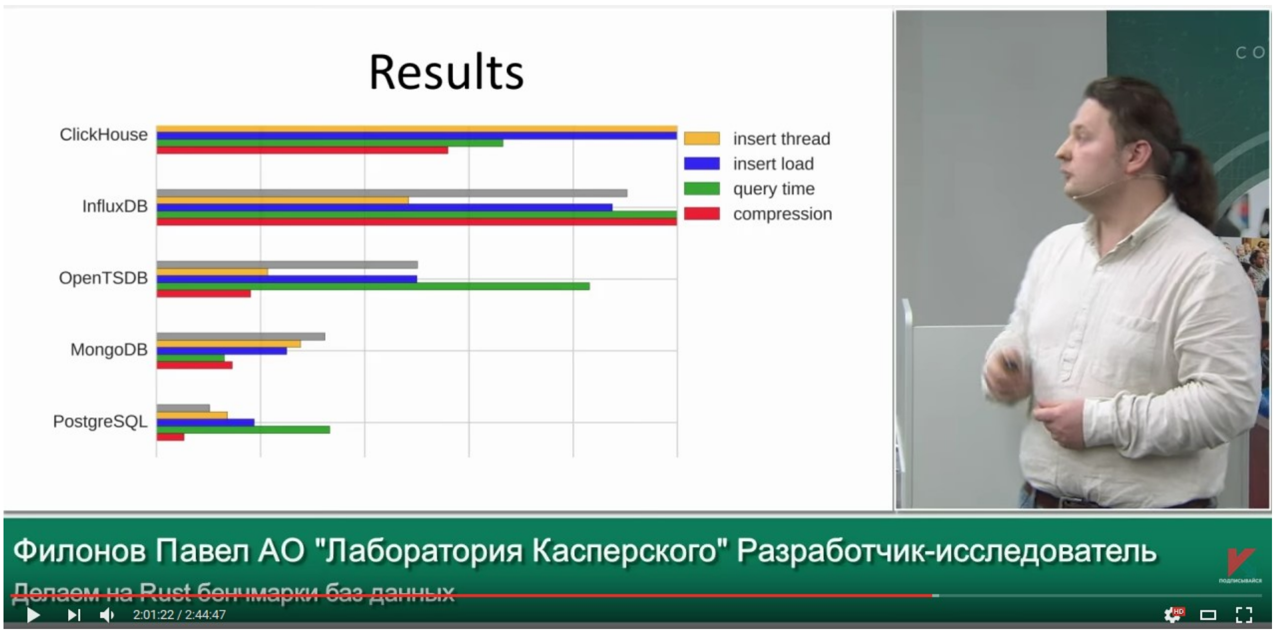
+ meetups. Moscow, Saint-Petersburg, Novosibirsk, Ekaterinburg, Minsk, Nizhny Novgorod, Berlin, Palo Alto... Beijing.

欧阳辰

https://zhuanlan.zhihu.com/p/22165241

Thank you. Questions.

ClickHouse for sensor data



ClickHouse vs. Greenplum

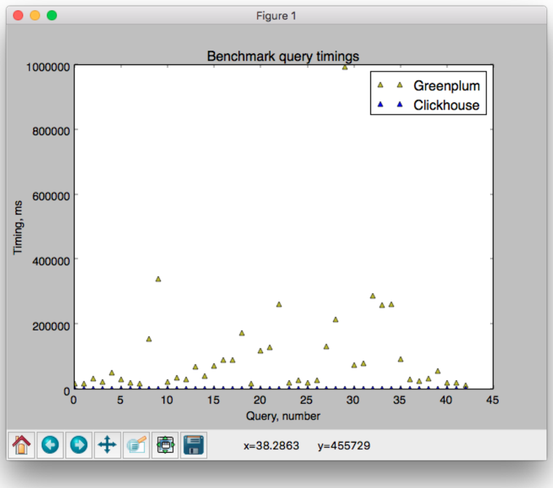


Figure 1
Benchmark query timings
Greenplum
Clickhouse
Timing, ms
Query, number
x=38.2863 y=455729