Supercharge Your Analytics with ClickHouse

Webinar

September 14th, 2017

Vadim Tkachenko
CTO, Percona
Alexander Zaitsev
CTO, Altinity

PERCONA

Analytic database landscape

Commercial solutions – fast and expensive

- Vertica
- RedShift
- Teradata
  - Etc
- The cost scales with your data

Open Source: somewhat slow, sometime buggy.
But free

- InfiniDB (now MariaDB ColumnStore)
- InfoBright
- GreenPlum (started as commerical)
- Hadoop systems
- Apache Spark

PERCONA

ClickHouse – fast and free!

OpenSourced in Jun 2016

ClickHouse story

Yandex.ru - Russian search engine

Yandex Metrika - Russian “Google Analytics”

Interactive Ad Hoc reports at multiple petabytes

- *30+ billions of events daily*

No commercial solution would be cost effective and no OpenSource solution to handle this scale.

That’s how ClickHouse was born

PERCONA

ClickHouse is extremely fast and scalable.

"We had no choice, but make it fast" by ClickHouse developers

Initial Requirements

- Fast. Really fast
- Data processing in real time
- Capable of storing petabytes of data
- Fault-tolerance in terms of datacenters
- Flexible query language

Technical details

Vectorized processing

Massively Parallel Processing

Shared nothing

Column store with late materialization (like C-Store and Vertica):

- Data compression
- Column locality
- No random reads

(more in details, in Russian, https://clickhouse.yandex/presentations/meetup7/internals.pdf)

Vectorized processing

Data is represented as small single-dimensional arrays (vectors), easily accessible for CPUs.

The percentage of instructions spent in interpretation logic is reduced by a factor equal to the vector-size

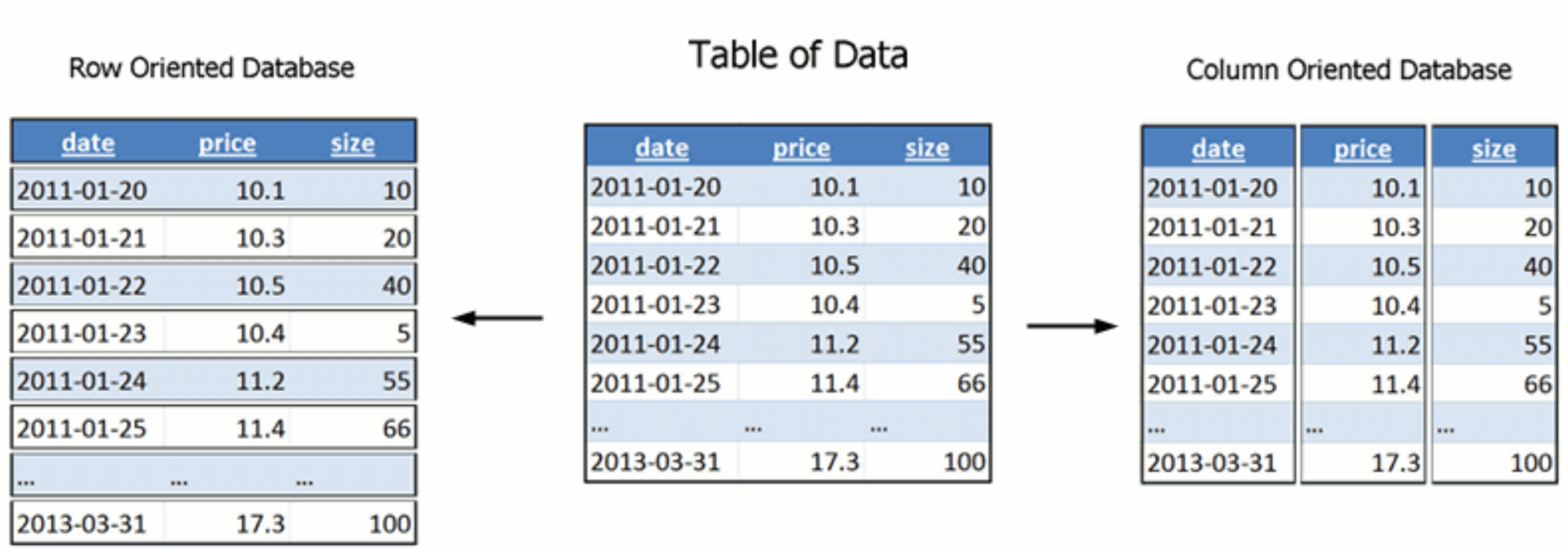
The functions that perform work now typically process an array of values in a tight loop

Tight loops can be optimized well by compilers, enable compilers to generate SIMD instructions automatically.

Modern CPUs also do well on such loops, out-of-order execution in CPUs often takes multiple loop iterations into execution concurrently, exploiting the deeply pipelined resources of modern CPUs.

It was shown that vectorized execution can improve data-intensive (OLAP) queries by a factor 50.

Column-oriented



date	price	size
2011-01-20	10.1	10
2011-01-21	10.3	20
2011-01-22	10.5	40
2011-01-23	10.4	5
2011-01-24	11.2	55
2011-01-25	11.4	66
...	...	...
2013-03-31	17.3	100

* The image taken from http://www.timestored.com/time-series-data/what-is-a-column-oriented-database

Efficient execution

```
SELECT Referer, count(*) AS count
FROM hits
WHERE CounterID = 1234 AND Date >= today() - 7
GROUP BY Referer
ORDER BY count DESC LIMIT 10
```

(* example from https://clickhouse.yandex/presentations/meetup7/internals.pdf)

- Vectorized processing
- Read only needed columns: CounterID, Referer, Date
- Compression
- With index (CounterID, Date) - fast discard of unneeded blocks

Single Server - MPP

Use multiple CPU cores on the single server

Real case: Apache log from the real web site – 1.56 billion records

Query:

```
SELECT extract(request_uri,'(w+)$') p,sum(bytes) sm,count(*) c
FROM apachelog
GROUP BY p
ORDER by c DESC limit 100
```

Query is suited for parallel execution – most time spent in extract function

Execution on single server

56 threads / 28 cores | Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz

Query execution time

With 1 thread allowed: 823.646 sec ~ 1.89 mln records/sec

With 56 threads allowed: 23.587 sec ~ 66.14 mln records/sec

Speedup: 34.9x times

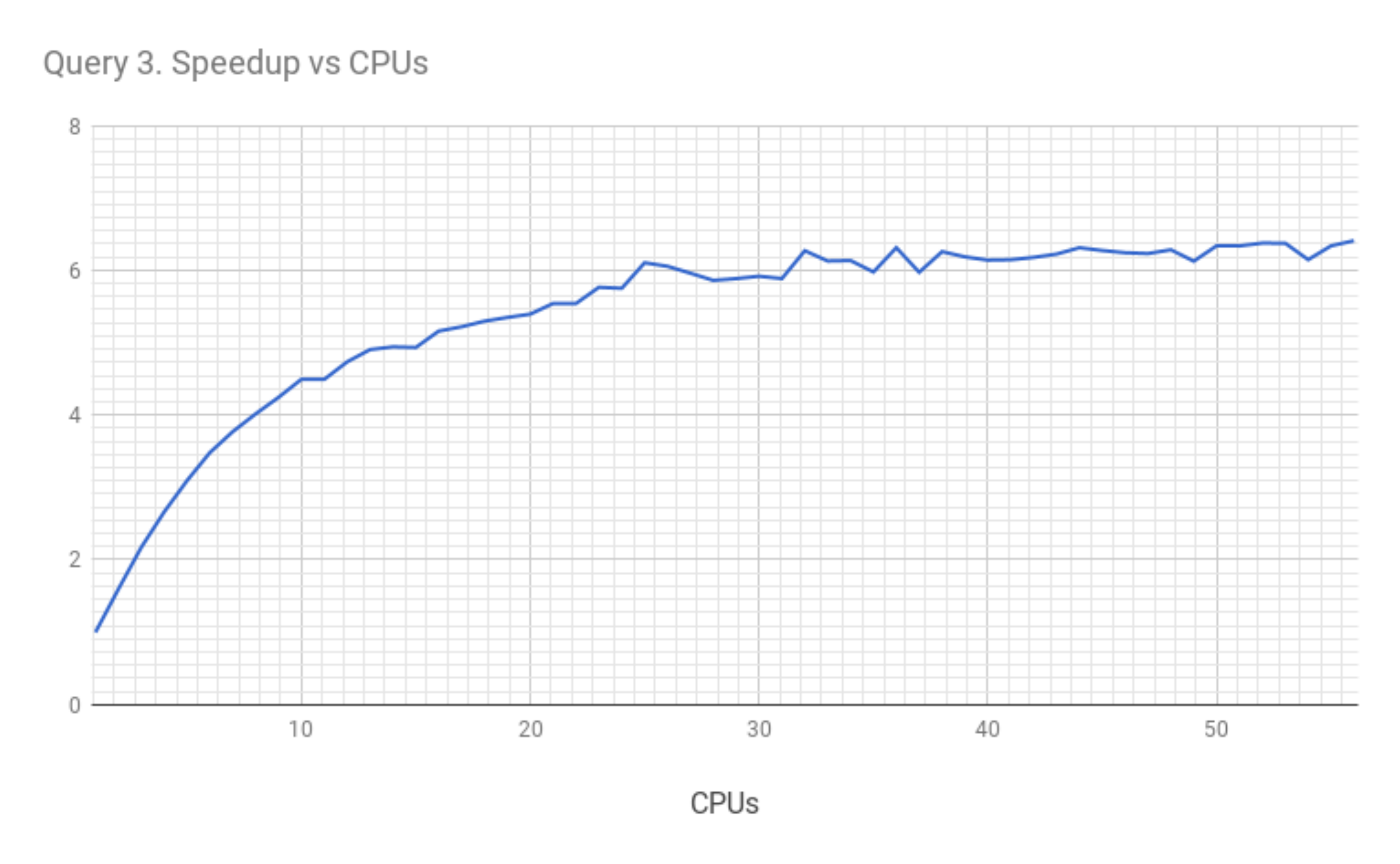

Query 1. Speedup vs CPUs
40
30
20
10
0
10
20
30
40
50
CPUs

Query 3

```
SELECT y, request_uri, cnt
FROM ( SELECT access_date y, request_uri, count(*) AS cnt
       FROM apachelog
                    GROUP BY y, request_uri
                    ORDER BY y ASC )
ORDER BY y,cnt DESC LIMIT 1 BY y
```

Less suitable for parallel execution – serialization to build a temporary table for internal subquery

Speedup: 6.4x times



Query 3. Speedup vs CPUs
8
6
4
2
0
10
20
30
40
50
CPUs

More details in the blog post:
https://www.percona.com/blog/2017/09/13/massive-parallel-log-processing-clickhouse/

PERCONA

Data distribution

If a single server is not enough

Distributed query

SELECT foo FROM distributed_table

SELECT foo FROM local_table GROUP BY col1

- Server 1

SELECT foo FROM local_table GROUP BY col1

- Server 2

SELECT foo FROM local_table GROUP BY col1

- Server 3

NYC taxi benchmark

CSV 227 GB, ~1.3 bln rows

SELECT passenger_count, avg(total_amount) FROM trips GROUP BY passenger_count

N Servers	1	3	140
Time, sec	1.224	0.438	0.043
Speedup		x2.8	x28.5

* Taken from https://clickhouse.yandex/presentations/meetup7/internals.pdf

PERCONA

Reliability

Any number of replicas

Any replication topology

Multi-master

Cross-DC

Asynchronous (for speed)

- ➔ *Delayed replicas, possible stale data reads*
- More on data distribution and replication https://www.altinity.com/blog/2017/6/5/clickhouse-data-distribution

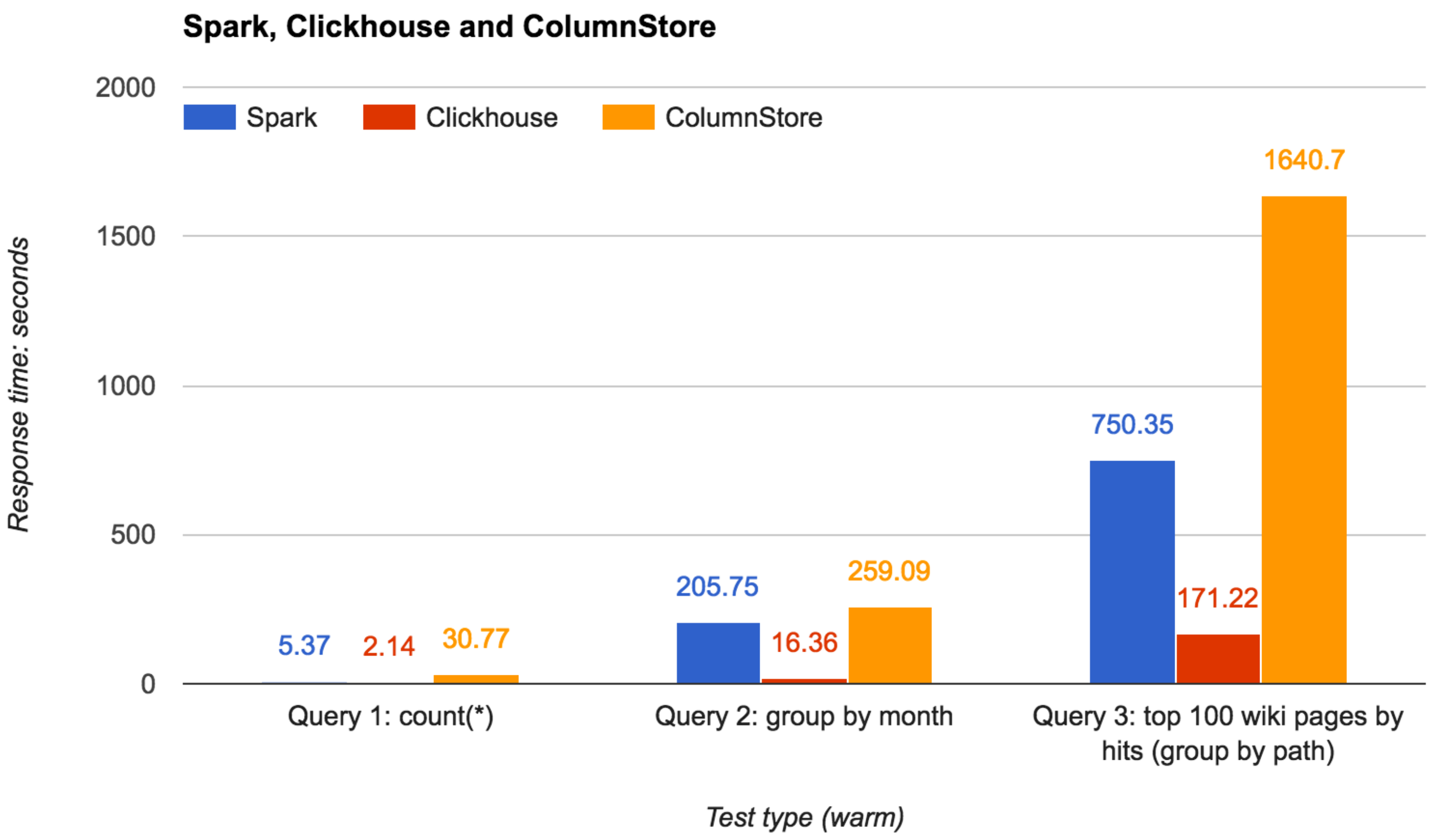
Benchmarks!

ClickHouse vs Spark vs MariaDB ColumnStore

Wikipedia page Counts, loaded full with the year 2008, ~26 billion rows

https://www.percona.com/blog/2017/03/17/column-store-database-benchmarks-mariadb-columnstore-vs-clickhouse-vs-apache-spark/

ClickHouse vs Spark vs MariaDB ColumnStore



Cloud: ClickHouse vs RedShift

https://www.altinity.com/blog/2017/6/20/clickhouse-vs-redshift

5 queries based on NYC taxi dataset

Query 1:
```
SELECT dictGetString('taxi_zones', 'zone', toUInt64(pickup_location_id)) AS zone, count() AS c
    FROM yellow_tripdata_staging
    GROUP BY pickup_location_id
    ORDER BY c DESC LIMIT 10
```

RedShift 1 instance / 3 instances of ds2.xlarge (4 vCPU / 31 GiB memory)

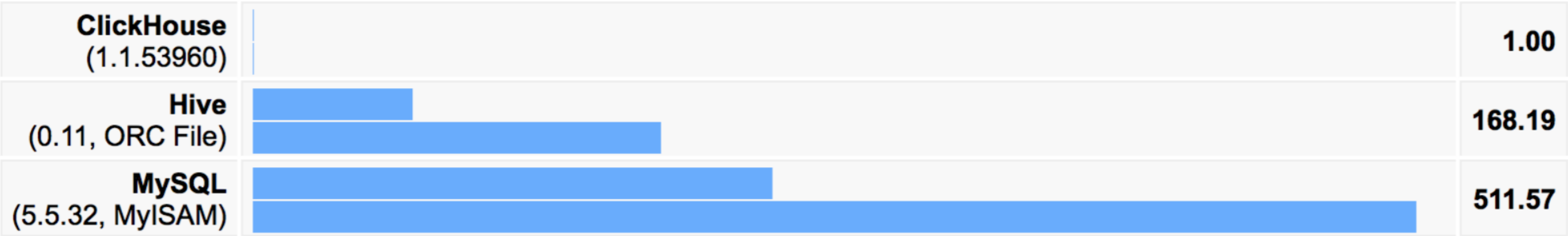
ClickHouse 1 instance r4.xlarge (4 vCPU / 30.5 GiB memory)


RedShift x1, RedShift x3 and CH on AWS x1
150
100
sec
50
0
query
15
7
3.3
1
0.5
8
47
20
31
140
48
36
70
20
32
1
2
3
4
5
RedShift x1
RedShift x3
CH on AWS x1

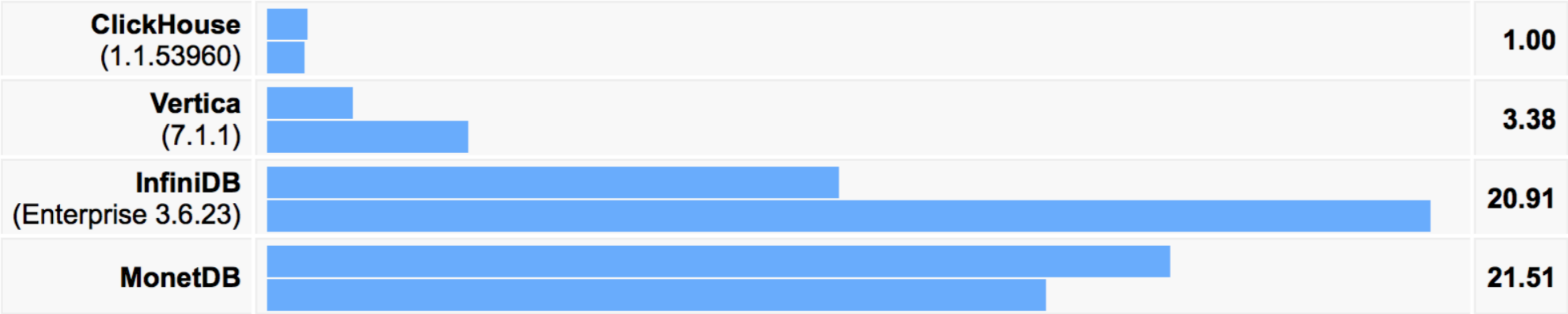
PERCONA

By Yandex, see [2]

Relative query processing time (lower is better):



Relative query processing time (lower is better):



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

ClickHouse – use cases

Adv networks data

Web/App analytics

Ecommerce/Telecom logs

Online games

Sensor data

Monitoring

PERCONA

ClickHouse – wrong cases

Not an OLTP

Not a key-value store

Not a document store

No UPDATEs/DELETEs – does not support data modification

ClickHouse - limitations

Custom SQL dialect

As a consequence -- limited ecosystem (can not fit to standard one)

No deletes/updates:

- but there are mutable table types (engines)
- there is a way to connect to external updatable data (dictionaries)

Somewhat hard to manage for now - no variety of tools to work with

Somewhat young

Resources for users

The Documentation is available in English!

- https://clickhouse.yandex/docs/en/

GUI Tool

- http://tabix.io

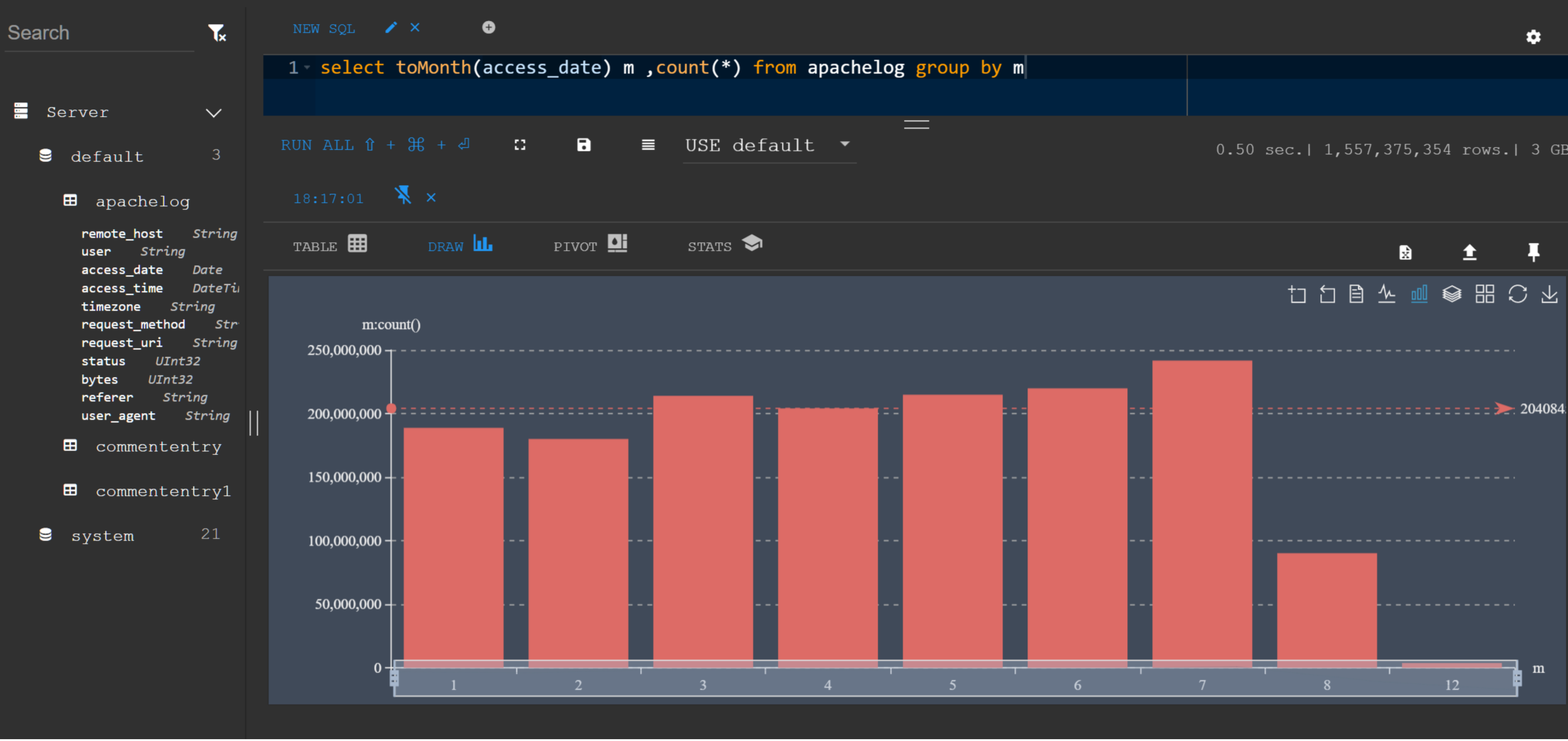
Apache Superset https://superset.incubator.apache.org supports ClickHouse

- a modern, enterprise-ready business intelligence web application

Grafana integration

- https://grafana.com/plugins/vertamedia-clickhouse-datasource

ODBC & JDBC drivers available



Search
NEW SQL
select toMonth(access_date) m ,count(*) from apachelog group by m
Server
default 3
apachelog
remote_host String
user String
access_date Date
access_time DateTi
timezone String
request_method Str
request_uri String
status UInt32
bytes UInt32
referer String
user_agent String
commententry
commententry1
system 21
RUN ALL
USE default
0.50 sec.| 1,557,375,354 rows.| 3 GB
18:17:01
TABLE
DRAW
PIVOT
STATS
m:count()
250,000,000
200,000,000
150,000,000
100,000,000
50,000,000
0
204084
1
2
3
4
5
6
7
8
12
m

PERCONA

Who is using ClickHouse?

Well, beside Yandex

Carto

- https://carto.com/blog/inside/geospatial-processing-with-clickhouse/

Percona

- We integrate ClickHouse as part of our Percona Monitoring and Management software

CloudFlare

- https://blog.cloudflare.com/how-cloudflare-analyzes-1m-dns-queries-per-second/

ClickHouse at CloudFlare

33 Nodes

8M+ inserts/sec

2PB+ disk size

More on CloudFlare experience

- https://www.altinity.com/sfmeetup2017

ClickHouse Demo on MemCloud

Kodiak Data and Altinity now Offer a Cloud Version of ClickHouse

ClickHouse

MEMCLOUD

Altinity

1. FASTEST MPP Open Source DBMS
2. Cutting Edge Cloud for Big Data Apps and Processing
3. World-class ClickHouse Expertise

[Try the ClickHouse on MemCloud demo here](http://clickhouse-demo.memcloud.works/)

http://clickhouse-demo.memcloud.works/

KODIAK DATA

Final words

Simply try it for your Analytics/Big Data case!

Need more info - http://clickhouse.yandex

My Contact: Vadim@percona.com

- *@VadimTk*

Get Your Tickets for Percona Live Europe!

Championing Open Source Databases

- MySQL, MongoDB, Open Source Databases
- Time Series Databases, PostgreSQL, RocksDB
- Developers, Business/Case Studies, Operations
- September 25-27th, 2017
- Radisson Blu Royal Hotel, Dublin, Ireland

OPEN SOURCE DATABASE CONFERENCE
PERCONA
LIVE EUROPE
DUBLIN

Last Year's Conference Sold Out!

Reserve your spot ASAP.

Talk to Percona Experts at AWS Re:Invent!

Database Performance for Cloud Deployments

- Percona Support and Managed Services
  - *Amazon RDS, Aurora, Roll Your Own*
  - *MySQL/MariaDB/MongoDB*
  - *Reduce costs and optimize performance*
- Percona Monitoring and Management Demos
  - *Point-in-time visibility and historical trending of database performance*
  - *Detailed query analytics*
- Booth #1138

ClickHouse Webinar

Alexander Zaitsev

LifeSteet, Altinity

Who am I

- Graduated Moscow State University in 1999
- Software engineer since 1997
- Developed distributed systems since 2002
- Focused on high performance analytics since 2007
- Director of Engineering in LifeStreet
- Co-founder of Altinity

Agenda

- LifeStreet ClickHouse implementation experience
- MySQL and ClickHouse

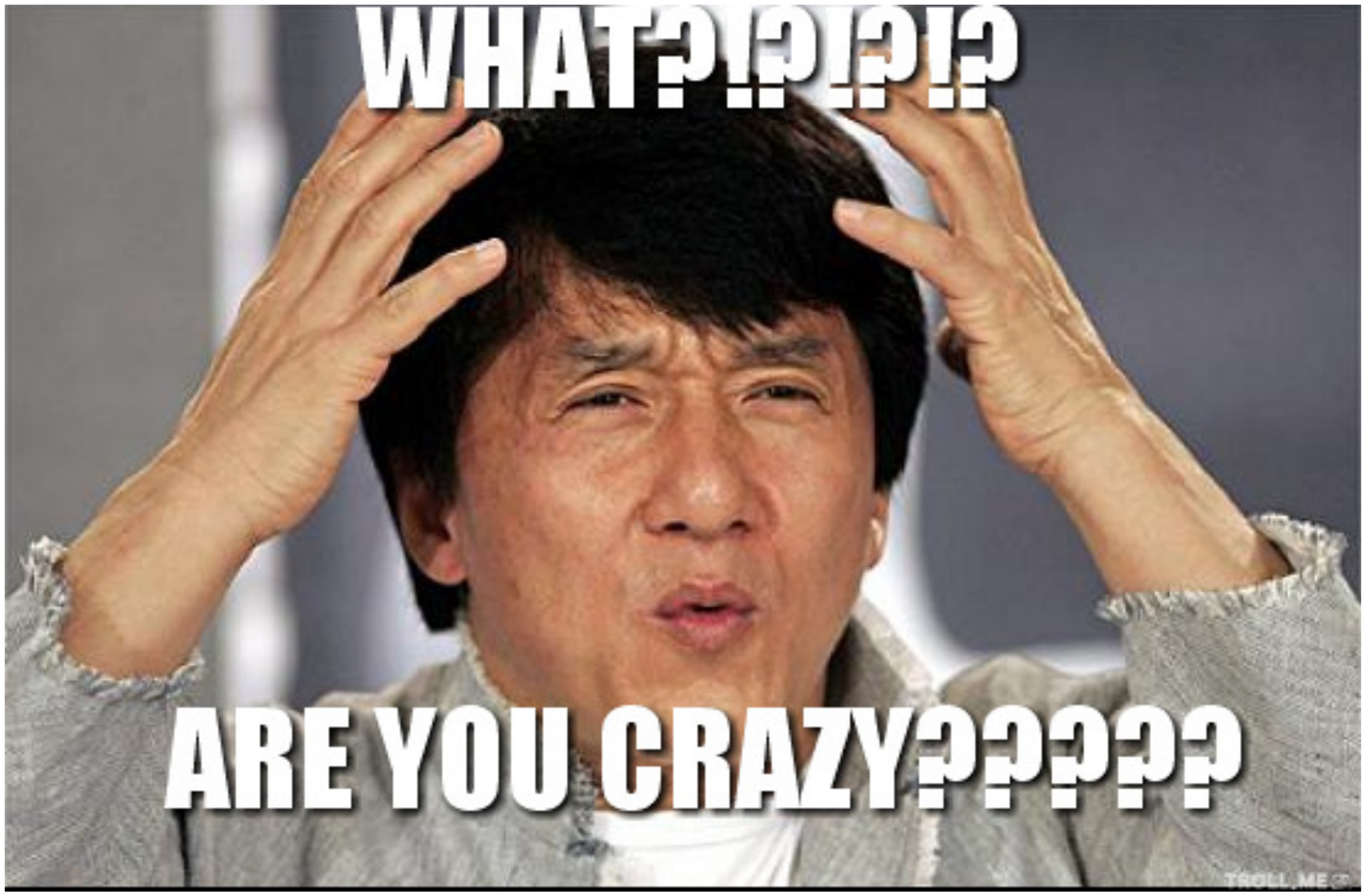
LIFESTREET

- Ad Tech company (ad exchange, ad server, RTB, DSP, DMP) since 2006
- 10,000,000,000+ events/day
- 10+ fact tables, 500+ dimensions, 100+ metrics
- Internal and external users, algos, MLs
- Different solutions tried and used in different years, including MySQL, Oracle, Vertica, many internal POCs
- Now -- ClickHouse

Flashback: ClickHouse at 08/2016

- 1-2 months in Open Source
- Internal Yandex product – no other installations
- No support, roadmap, communicated plans
- 3 official devs
- A number of visible limitations (and many invisible)
- Stories of other doomed open-sourced DBs

Develop production system with “that”?



ClickHouse is/was missing:

- Transactions
- Constraints
- Consistency
- UPDATE/DELETE
- NULLs (not anymore)
- Milliseconds
- Implicit type conversions
- Full SQL support
- Partitioning by any column (date only)
- Cluster management tools

But we tried and succeeded

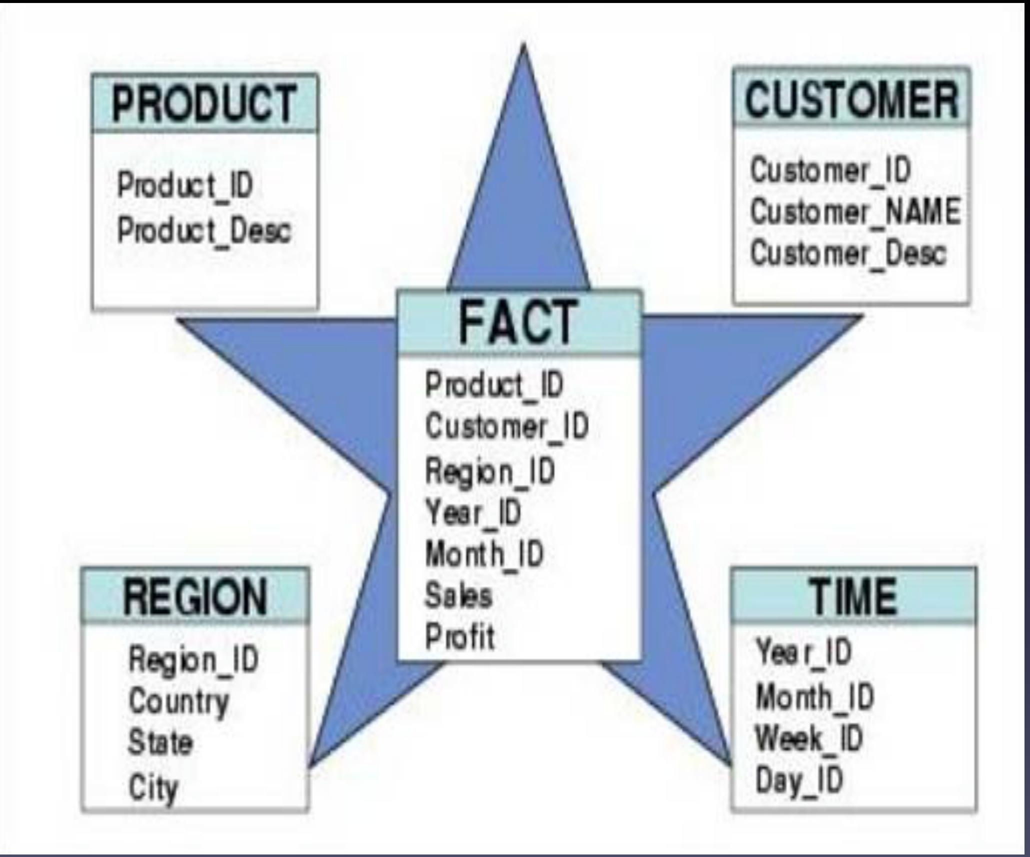
Migration problem: basic things do not fit

Main Challenges

- Design efficient schema
  - Use ClickHouse bests
  - Workaround limitations
- Design sharding and replication
- Reliable data ingestion
- Client interfaces

Typical schema: “star”

- Facts
- Dimensions
- Metrics
- Projections



De-normalized vs. normalized

De-normalized (dimensions in fact table):

- Easy
- Simple queries
- No data changes are possible
- Sub-efficient storage
- Sub-efficient queries

Normalized (dimensions in separate tables):

- More difficult to maintain
- More complex queries
- Dimensions can change
- More efficient storage
- More efficient queries

Normalized schema: traditional approach - joins

- Limited support in ClickHouse (1 level, cascade sub-selects for multiple)
- Dimension tables are not updatable

Dictionaries - ClickHouse dimensions approach

- Lookup service: key -> value
- Supports multiple external sources (files, databases etc.)
- Refreshable

Dictionaries. Example

```
SELECT country_name,
       sum(imps)
  FROM T
  ANY INNER JOIN dim_geo USING (geo_key)
 GROUP BY country_name;

vs

SELECT dictGetString(‘dim_geo’, ‘country_name’, 
geo_key) country_name,
       sum(imps)
  FROM T
 GROUP BY country_name;
```

Dictionaries. Configuration

```
<dictionary>
    <name></name>
    <source> … </source>
    <lifetime> ... </lifetime>
    <layout> … </layout>
    <structure>
        <id> ... </id>
        <attribute> ... </attribute>
        <attribute> ... </attribute>
        ...
    </structure>
</dictionary>
```

Dictionaries. Sources

- file
- mysql table
- clickhouse table
- odbc data source
- executable script
- http service

Dictionaries. Layouts

- flat
- hashed
- cache
- complex_key_hashed
- range_hashed

Dictionaries. range_hashed

- 'Effective Dated' queries

```
<layout>
    <range_hashed />
</layout>
<structure>
    <id>
        <name>id</name>
    </id>
    <range_min>
        <name>start_date</name>
    </range_min>
    <range_max>
        <name>end_date</name>
    </range_max>
```

```
dictGetFloat32('srv_ad_serving_costs',
'ad_imps_cpm', toUInt64(0), event_day)
```

Dictionaries. Update values

- By timer (default)
- Automatic for MySQL MyISAM
- Using ‘invalidate_query’
- Manually touching config file
- N dict * M nodes = N * M DB connections

Dictionaries. Restrictions

- ‘Normal’ keys are only UInt64
- No on demand update (added in 1.1.54289)
- Every cluster node has its own copy
- XML config (DDL would be better)

Tables

- Engines
- Sharding
- Distribution
- Replication

Engine = ?

- In memory:
  - Memory
  - Buffer
  - Join
  - Set
- On disk:
  - Log, TinyLog
  - MergeTree family
- Virtual:
  - Merge
  - Distributed
  - Dictionary
  - Null
- Special purpose:
  - View
  - Materialized View

Merge tree

- What is ‘merge’
- PK sorting
- Date partitioning
- Query performance

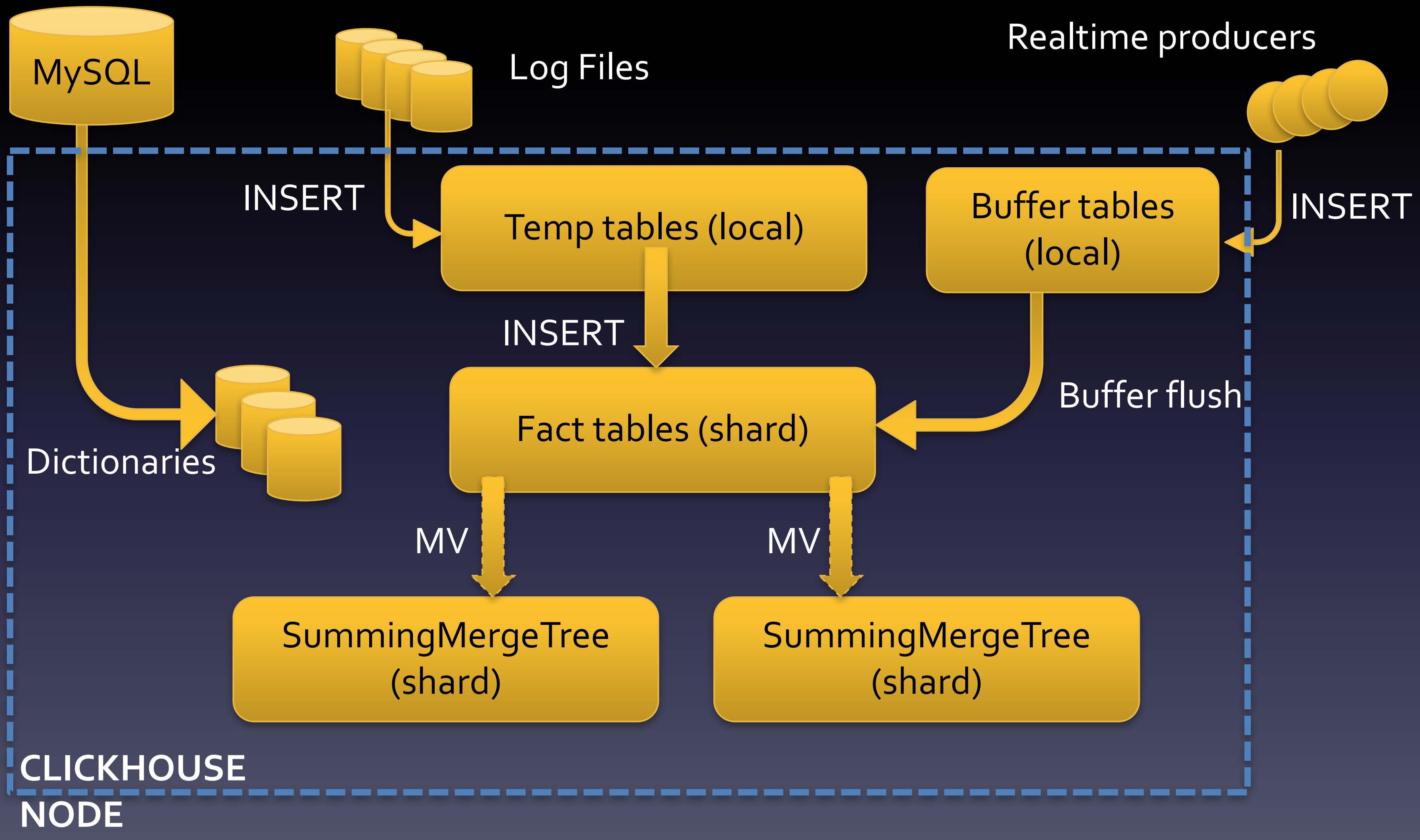
Data Load

- Multiple formats are supported, including CSV, TSV, JSONs, native binary
- Error handling
- Simple Transformations
- Load locally (better) or distributed (possible)
- Temp tables help
- Replicated tables help with de-dup

The power of Materialized Views

- MV is a table, i.e. engine, replication etc.
- Updated synchronously
- SummingMergeTree – consistent aggregation
- Alters are not straightforward, but possible

Data Load Diagram



Updates and deletes

- Dictionaries are updatable
- Replacing and Collapsing merge trees
  - eventually updates
  - SELECT … FINAL
- Partitions

Sharding and Replication

- Sharding and Distribution => Performance
  - Fact tables and MVs – distributed over multiple shards
  - Dimension tables and dicts – replicated at every node (local joins and filters)
- Replication => Reliability
  - 2-3 replicas per shard
  - Cross DC

SQL

- Supports basic SQL syntax
- Non-standard JOINs implementation:
  - 1 level only
  - ANY vs ALL
  - only USING
- Aliasing everywhere
- Array and nested data types, lambda-expressions, ARRAY JOIN
- GLOBAL IN, GLOBAL JOIN
- Approximate queries
- TopX support (LIMIT N BY)

Main Challenges Revisited

- Design efficient schema
  - Use ClickHouse bests
  - Workaround limitations
- Design sharding and replication
- Reliable data ingestion
- Client interfaces

Migration project timelines

- August 2016: POC
- October 2016: first test runs
- December 2016: production scale data load:
  - 10-50B events/ day, 20TB data/day
  - 12 x 2 servers with 12x4TB RAID10
- March 2017: Client API ready, starting migration
  - 30+ client types, 20 req/s query load
- May 2017: extension to 20 x 3 servers
- June 2017: migration completed

ClickHouse at fall 2017

- 1+ year Open Source
- 100+ prod installs worldwide
- Public changelogs, roadmap, and plans
- 10+ devs, community contributors
- Active community, blogs, case studies
- A lot of features added by community requests
- Support by Altinity

So now it is much easier

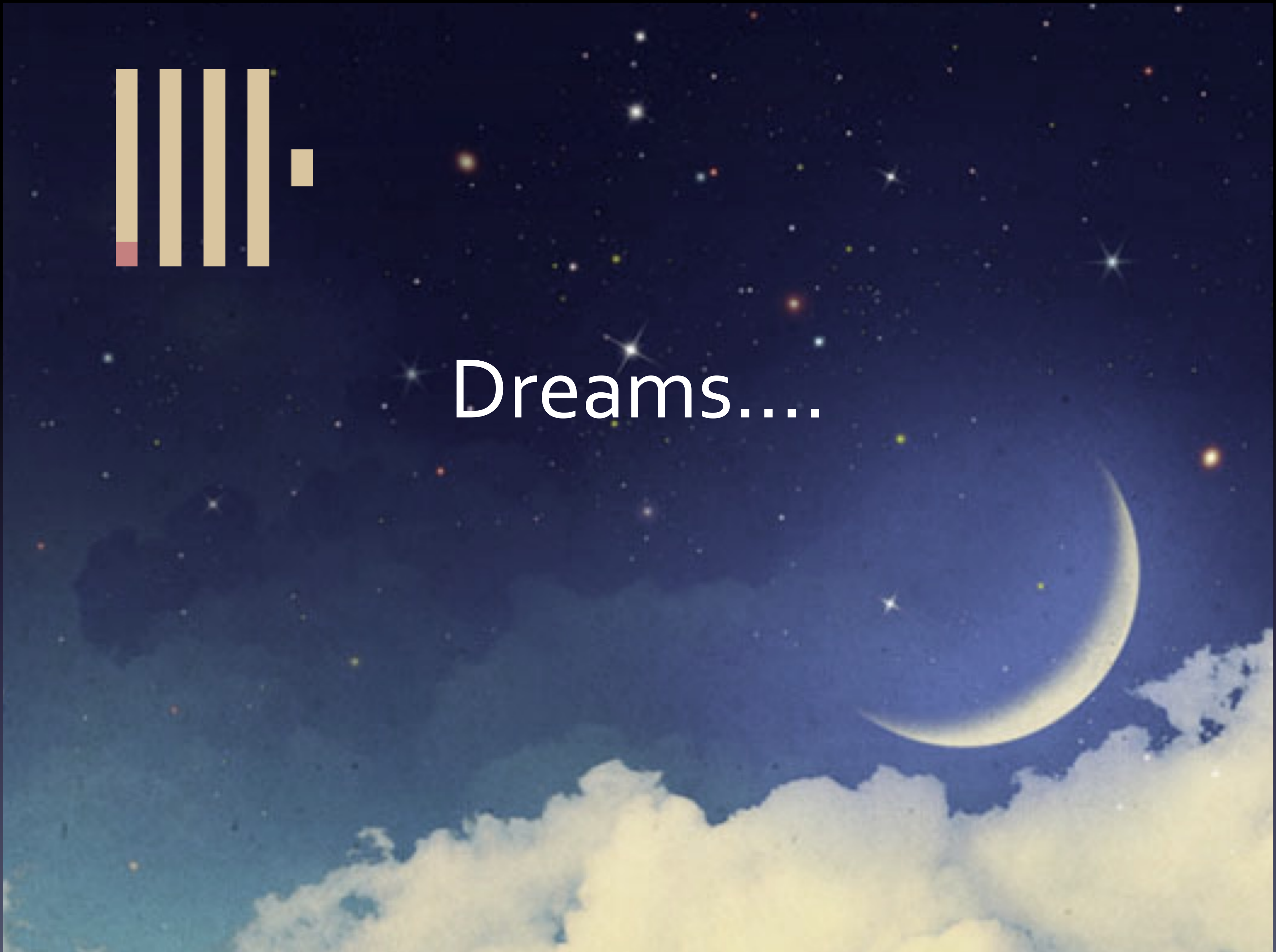
ClickHouse and MySQL

- MySQL is widespread but weak for analytics
  - TokuDB, InfiniDB somewhat help
- ClickHouse is best in analytics

How to combine?

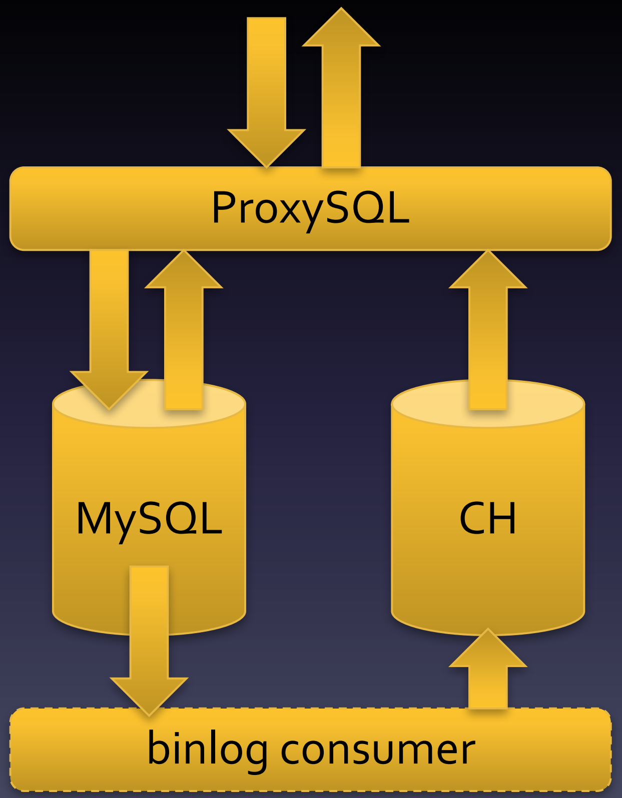
Imagine

MySQL flexibility at ClickHouse speed?


Dreams....

ClickHouse *with* MySQL

- ProxySQL to access ClickHouse data via MySQL protocol (already available)
- Binlogs integration to load MySQL data in ClickHouse in realtime (in progress)



ClickHouse *instead of* MySQL

- Web logs analytics
- Monitoring data collection and analysis
  - Percona's PMM
  - Infinidat InfiniMetrics
- Other time series apps

Questions?

Contact me:

alexander.zaitsev@lifestreet.com
alz@altinity.com
skype: alex.zaitsev