ClickHouse as Time-Series Database for Graphite

Dmitry Andreev, Yandex
About me

› Software Engineer
› Head of Infrastructure Development Group in Yandex Market
› AndreevDm@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
History
End of 2014

- Couple of graphite installations is Yandex
- We use installations on Ceres
- We intensively add new metrics
History, part 2

- Beginning of 2015
  - The graphite installation is overloaded
  - We are launching a prototype on ClickHouse

- April 2015
  - First Graphouse commit
ClickHouse briefly

- Open-source column-oriented DBMS
- Linearly scalable
- Blazingly fast
- SQL dialect with extensions
Yandex.Metrica

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
Why ClickHouse?
Time Series Database Requirements

1. ...
Time Series Database Requirements

1. High write performance
2. Fast range queries
3. Data compaction (rollup)
Table consists of partitions

Partition is sorted by Primary Key (PK is not unique)
ClickHouseMergeTreeTable Engine

- Table consists of partitions
- Partition is sorted by Primary Key (PK is not unique)
- Insert creates new partition
ClickHouse MergeTree Table Engine

- Primary Key - (Metric, Timestamp)
- Time Series Database

Requirements:
- 1. Fast range queries
- 2. High write performance

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Timestamp</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>metric1</td>
<td>1490000100</td>
<td>42.42</td>
</tr>
<tr>
<td>metric1</td>
<td>1490000160</td>
<td>42.21</td>
</tr>
<tr>
<td>metric1</td>
<td>1490000220</td>
<td>21.42</td>
</tr>
<tr>
<td>metric1</td>
<td>1492383280</td>
<td>1</td>
</tr>
<tr>
<td>metric2</td>
<td>1490000100</td>
<td>1.1</td>
</tr>
<tr>
<td>metric2</td>
<td>1490000160</td>
<td>1.2</td>
</tr>
<tr>
<td>metric2</td>
<td>1490000220</td>
<td>1.3</td>
</tr>
<tr>
<td>metric3</td>
<td>1490000100</td>
<td>5</td>
</tr>
<tr>
<td>metric3</td>
<td>1490000220</td>
<td>3.3</td>
</tr>
<tr>
<td>metric3</td>
<td>1490000340</td>
<td>1.33333</td>
</tr>
</tbody>
</table>
GraphiteMergeTree Table Engine

- Compacting data during the merge
ClickHouse as Time-Series Database for Graphite

Who the **** is Mr. Graphouse?
Common Graphite Architecture

![Diagram of Graphite Architecture]

Legend:
- **logical component**: required
- **optional**

Components:
- **Graphite-Web**
  - `webapp`
  - `memcached`
  - `database`
- **Carbon**
  - `carbon-relay`
  - `carbon-aggregator`
  - `carbon-cache`
- **Whisper**
  - `load.1m.wsp`
  - `load.5m.wsp`
  - `load.15m.wsp`
Graphite query workflow

1. \( \text{avg(servers.clickhouse0{1,2}.disk*.util), period} \) → Graphite-web

2. Graphite-web → servers.clickhouse0{1,2}.disk*.util → StorageFinder

3. Graphite-web → Points → StorageReader

4. Graphite-web → (apply functions, render image, …)
Metric Tree

ROOT

servers
  - clickhouse01
  - clickhouse02
  - clickhouse03

applications
  - app1
  - app2
  - app3
    - BAN

disk-sda1
  - free

disk-sda2
  - util

cpu
  - idle
  - user

In memory

On demand
(with cache)
Architecture with Graphouse

Original Graphite architecture

Graphite-Web
- webapp
- memcached
- database

Carbon
- carbon-relay
- carbon-aggregator
- carbon-cache

Whisper
- load1m.wsp
- load5m.wsp
- load15m.wsp

Graphhouse architecture

Graphite-Web
- webapp
- memcached
- database

ClickHouse

Graph house
ClickHouse as Time-Series Database for Graphite

Sharding and replication
ClickHouse Cluster Scheme

Data Center

Shard
Graphouse replications

- Metric tree synchronisation via ClickHouse Table
- Eventual Consistency
Graphite cluster. Case 1.
Graphite cluster. Case 2.

Graphite-web

Graphouse

Load

ClickHouse

ClickHouse

ClickHouse

ClickHouse
Graphite cluster. Case 3.
Our experience

- SSD is good, but HDD is fine too
- Don’t use sharding unless you really need
ClickHouse as Time-Series Database for Graphite

Benchmarks
Candidates

1. Carbon + Whisper
2. Go-carbon + Whisper
3. Biggraphite + Cassandra
4. Graphouse + Clickhouse
Hardware

- Intel(R) Xeon(R) CPU E5-2650 v2 (2.60 GHz, 8 Cores, 16 Threads)
- 128 GB RAM
- 2 SSD in RAID1 (Limited by SATA-300)
Insert test

- Storage schema: 60s:1d,10m:30d
- Send interval: 30 seconds
- Metrics per thread: 30 000
- Send timeout - 30 seconds
- Thread count - variable (average 1000 mps per thread)
Benchmark Util

andreevdm@host:~/$ inputPerfTest --threads 2000 --metrics 30000 --interval 30 --prefix perf-test. --runMinutes 600

2017-04-27 00:25:05,199 INFO  [InsertTest main] Creating graphite insert perf test for localhost:2003
2017-04-27 00:25:05,261 INFO  [InsertTest main] Test run time is 600 minutes
...
2017-04-27 00:26:05,815 INFO  [InsertTest pool-2-thread-20] Finished sending metrics for timestamp 1493241935 (Thu Apr 27 00:25:35 MSK 2017). Send 60000000 (61290000 total), failed to send 0 (0 total), errors 0.0% (0.0% total). Total send time 74525 ms, avg
Insert performance (metrics per second)

Graphouse + ClickHouse

Carbon + Wisper

Go-carbon + Wisper

Biggraphite + Casandra

2,000,000

95,000

1,050,000

45,000
What’s next?

ClickHouse as Time-Series Database for Graphite
Graphite query workflow

1. \( \text{avg(servers.clickhouse0\{1,2\}.disk*.util), period} \rightarrow \text{Graphite-web} \)

2. \( \text{servers.clickhouse0\{1,2\}.disk*.util} \leftarrow \text{Graphite-web} \rightarrow \text{StorageFinder} \)

3. \( \text{Graphite-web} \leftarrow \text{Points} \rightarrow \text{StorageReader} \)

4. \( \text{Graphite-web} \rightarrow (\text{apply functions, render image, ...}) \)
Query Example

```
| avg(servers.clickhouse0{1,2}.disk*.util) |
```

```
SELECT metric, timestamp, value
FROM graphite.data
WHERE metric IN ( 
    'servers.clickhouse01.disk-sda1.util', 'servers.clickhouse01.disk-sda2.util', 
    'servers.clickhouse02.disk-sda1.util', 'servers.clickhouse02.disk-sda2.util'
) AND DATE ...
```
We can do better

```
avg(servers.clickhouse0{1,2}.disk*.util)
```

```
SELECT timestamp, avg(value)
FROM graphite.data
WHERE metric IN ('servers.clickhouse01.disk-sda1.util', 'servers.clickhouse01.disk-sda2.util',
                 'servers.clickhouse02.disk-sda1.util', 'servers.clickhouse02.disk-sda2.util')
AND DATE ...
GROUP BY timestamp
```
Notice:  
ClickHouse is not TSDB  
It can do much more
Rate My Session!

Schedule
Timezone: Europe/Berlin +02:00

11:20

Introducing gh-ost: triggerless, painless, trusted online schema migrations
11:20 → 12:10, Matterhorn 2

Rate & Review
Tap to rate

Feedback (optional)

SUBMIT
Contacts

Dmitry Andreev
Head of Infrastructure Development Group in Yandex Market

✉️ AndreevDm@yandex-team.ru
皆さん
https://github.com/yandex/graphouse (Apache License 2.0)
✍️ https://t.me/graphouse