



GRAPHITE + CLICKHOUSE

---

*История одного переезда*



# GRAPHITE ROLLUP

# GRAPHITE ROLLUP

## Правила rollup в Graphite:

## storage-schemas.conf:

[default]

```
pattern = .*
```

retentions = 60s:14d,5m:45d,1h:2y

## storage-aggregation.conf:

[default]

```
pattern = .*
```

xFilesFactor = 0.5

```
aggregationMethod = average
```

## Правила rollup в ClickHouse:

```
[
  'pattern' => [
    { 'regexp' => ['^(some).*'],
      'function' => ['avg'],
      'retention' => [
        { 'age' => [0], 'precision' => [60], },
        { 'age' => [1209600], 'precision' => [300], },
        { 'age' => [3888000], 'precision' => [3600], },
        { 'age' => [63072000], 'precision' => [86400], },
      ],
    },
  ],
  'default' => [
    { 'function' => ['avg'],
      'retention' => [
        { 'age' => [0], 'precision' => [60], },
        { 'age' => [3600], 'precision' => [86400], },
      ],
    },
  ],
]
```

# GRAPHITE ROLLUP

ClickHouse 19.3 и ранее:

```
'pattern' => [  
  { 'regexp' => ['^(some).*\.min$'],  
    'function' => ['min'],  
    'retention' => [  
      { 'age' => [0], 'precision' => [60], },  
      { 'age' => [1209600], 'precision' => [300], },  
      { 'age' => [3888000], 'precision' => [3600], },  
      { 'age' => [63072000], 'precision' => [86400], },  
    ],  
  },  
  { 'regexp' => ['^(some).*'],  
    'function' => ['avg'],  
    'retention' => [  
      { 'age' => [0], 'precision' => [60], },  
      { 'age' => [1209600], 'precision' => [300], },  
      { 'age' => [3888000], 'precision' => [3600], },  
      { 'age' => [63072000], 'precision' => [86400], },  
    ],  
  },  
],  
'default' => ['*'],
```

ClickHouse, начиная с 19.4:

```
'pattern' => [  
  { 'regexp' => ['\.min$'], 'function' => ['min'], },  
  { 'regexp' => ['^(some).*'],  
    'retention' => [  
      { 'age' => [0], 'precision' => [60], },  
      { 'age' => [1209600], 'precision' => [300], },  
      { 'age' => [3888000], 'precision' => [3600], },  
      { 'age' => [63072000], 'precision' => [86400], },  
    ],  
  },  
],  
'default' => [  
  { 'function' => ['avg'],  
    'retention' => [  
      { 'age' => [0], 'precision' => [60], },  
      { 'age' => [3600], 'precision' => [86400], },  
    ],  
  },  
]
```

# GRAPHITE ROLLUP CLIENTS: GRAPHHOUSE

- Поддерживается в master с 2019-08-08
- <https://github.com/yandex/graphouse/pull/110>



# GRAPHITE ROLLUP

## CLIENTS: GRAPHITE-CRICKHOUSE

- Поддерживается с версии 0.9.0, в мастер с 2019-05-16
- <https://github.com/lomik/graphite-clickhouse/commit/24285713a>





# ПЕРВЫЕ МЕТРИКИ

# ПЕРВЫЕ МЕТРИКИ

- ◆ На момент изменений:

- ◆ grafsy →

- ◆ carbon-c-relay →

- ◆ carbonwriter (grobian/carbonwriter)

- ◆ План:

- ◆ grafsy →

- ◆ carbon-c-relay ⇒

- ◆ → carbonwriter

- ◆ → graphouse





# ПЕРВЫЕ МЕТРИКИ

## ◆ На момент изменений:

- ◆ grafsy →
- ◆ carbon-c-relay →
- ◆ carbonwriter (grobian/carbonwriter)

## ◆ План:

- ◆ grafsy →
- ◆ carbon-c-relay ⇒
  - ◆ → carbonwriter
  - ◆ → graphouse

## Реальность:

- ◆ Graphouse архитектурно не может быстро прогреться при отправке метрик из ограниченного количества источников (параметр **-w**, workers carbon-c-relay) и перестаёт принимать метрики. При этом carbon-c-relay начинает их дропать при достижении лимита в параметре **-q**, queuesize (25к по умолчанию)
- ◆ Было несколько вариантов решения, но решили научить grafsy слать метрики в несколько приёмников

# ПЕРВЫЕ МЕТРИКИ: GRAFSY

- <https://github.com/innogames/grafsy>
  - Активный форк, основной репозиторий:  
<https://github.com/leoleovich/grafsy>
- Легкий демон, который:
  - Кеширует метрики локально
  - Отправляет метрики в несколько источников
  - Умеет читать метрики из файлов на диске
  - Имеет встроенную валидацию метрик
  - Написан на go






# ПЕРВЫЕ ЗАПРОСЫ В GRAPHITE-WEB

# ПЕРВЫЕ ЗАПРОСЫ: GRAPHHOUSE.PY

Текущая инфраструктура graphite-web:

- ◆ graphite-web 1.0.2
- ◆ python 2
- ◆ 3 сервера-шарда с whisper storage

План:

- ◆ graphite-web 1.1.x
  - ◆ python 3
  - ◆ ClickHouse кластер
  - ◆ Несколько серверов с graphouse
- 

# ПЕРВЫЕ ЗАПРОСЫ: GRAPHHOUSE.PY

Текущая инфраструктура graphite-web:

- ◆ graphite-web 1.0.2
- ◆ python 2
- ◆ 3 сервера-шарда с whisper storage

План:

- ◆ graphite-web 1.1.x
- ◆ python 3
- ◆ ClickHouse кластер
- ◆ Несколько серверов с graphhouse

Реальность:

- ◆ Плагин graphhouse.py не заработал ни с py3, ни с graphite 1.1.x
- ◆ В graphite-web с 1.0 по 1.1 изменился API fetch multiple points и алгоритм graphhouse.py из  $O=1$  превратился в  $O=n^2$
- ◆ Так как в компании принята стратегия "py3 only", было необходимо научить graphhouse работать с find\_multi корректно
- ◆ <https://github.com/yandex/graphhouse/pull/129>

# ПЕРВЫЕ ЗАПРОСЫ: SELECT FROM ()

```
SELECT
  metric,
  ts,
  avg(value) AS value
FROM
  ( SELECT
    metric,
    ts,
    argMax(value, updated) AS value
  FROM data
  WHERE (
    metric IN (.....) AND
    (ts >= 1563984700) AND (ts < 1564001100)
    AND (date >= toDate(1563984700))
    AND (date <= toDate(1564001100))
  )
  GROUP BY
    metric,
    timestamp AS ts )
GROUP BY
  metric,
  intDiv(toUInt32(ts), 60) * 60 AS ts
ORDER BY
  metric ASC,
  ts ASC
```

# ПЕРВЫЕ ЗАПРОСЫ: SELECT FROM ()

```
SELECT
  metric,
  ts,
  avg(value) AS value
FROM
  ( SELECT
    metric,
    ts,
    argMax(value, updated) AS value
  FROM data
  WHERE (
    metric IN (.....) AND
    (ts >= 1563984700) AND (ts < 1564001100)
    AND (date >= toDate(1563984700))
    AND (date <= toDate(1564001100))
  )
  GROUP BY
    metric,
    timestamp AS ts )
GROUP BY
  metric,
  intDiv(toUInt32(ts), 60) * 60 AS ts
ORDER BY
  metric ASC,
  ts ASC
```

Проблема: удалённо выполняется только подзапрос.

# ПЕРВЫЕ ЗАПРОСЫ: SELECT FROM (), РЕШЕНИЕ

```
CREATE VIEW graphite.data_view
(
    `metric` String,
    `value` Float64,
    `timestamp` UInt32,
    `date` Date
) AS
SELECT
    metric,
    timestamp,
    argMax(value, updated) AS value,
    date
FROM graphite.data_lr
GROUP BY
    metric,
    timestamp,
    date
```

```
CREATE TABLE graphite.data_view_distributed ...
```

```
SELECT
    metric,
    ts,
    avg(value) AS value
FROM
( SELECT
    metric,
    timestamp AS ts,
    value
FROM data_view_distributed
WHERE (
    metric IN (...)
    AND (ts >= 1563984700) AND (ts < 1564001100)
    AND (date >= toDate(1563984700))
    AND (date <= toDate(1564001100))
)
GROUP BY
    metric,
    intDiv(toUInt32(ts), 60) * 60 AS ts
ORDER BY
    metric ASC,
    ts ASC
```



# ПЕРВЫЕ ЗАПРОСЫ: SELECT FROM (), РЕШЕНИЕ

```
SELECT
  metric,
  ts,
  avg(value) AS value
FROM
  ( SELECT
    metric,
    ts,
    argMax(value, updated) AS value
  FROM data
  WHERE (
    metric IN (.....) AND
    (ts >= 1563984700) AND (ts < 1564001100)
    AND (date >= toDate(1563984700))
    AND (date <= toDate(1564001100))
  )
  GROUP BY
    metric,
    timestamp AS ts )
GROUP BY
  metric,
  intDiv(toUInt32(ts), 60) * 60 AS ts
ORDER BY
  metric ASC,
  ts ASC
```

```
SELECT
  metric,
  ts,
  avg(value) AS value
FROM
  ( SELECT
    metric,
    timestamp AS ts,
    value
  FROM data_view_distributed
  WHERE (
    metric IN (...)
    AND (ts >= 1563984700) AND (ts < 1564001100)
    AND (date >= toDate(1563984700))
    AND (date <= toDate(1564001100))
  )
  GROUP BY
    metric,
    intDiv(toUInt32(ts), 60) * 60 AS ts
ORDER BY
  metric ASC,
  ts ASC
```

# ПЕРВЫЕ ЗАПРОСЫ: SELECT FROM (), РЕШЕНИЕ

Было:

Peak memory usage (for query): 34.02 MiB.

Expression

MergeSorting

PartialSorting

Expression

Aggregating

Concat

Expression

Expression

Expression

MergingAggregated

Union

Materializing

ParallelAggregating

Expression × 10

Filter

MergeTreeThread

Remote × 3

Стало:

Peak memory usage (total): 8.28 MiB.

Expression

MergeSorting

PartialSorting

Expression

ParallelAggregating

Expression

Expression

Materializing

Expression

Filter

Materializing

Expression

Expression

ParallelAggregating

Expression × 4

Filter

MergeTreeThread

Expression × 3

Expression

Remote

# ПЕРВЫЕ ЗАПРОСЫ: SELECT FROM ()

- Полные результаты эксперимента:  
<https://gist.github.com/Felixoid/2837197691505b4c487a4dfb98982de0>
- PR в  
graphouse: <https://github.com/yandex/graphouse/pull/130>
- За помощь в решении спасибо Denny  
Crane [https://t.me/den\\_crane](https://t.me/den_crane)





## ДРУГИЕ ЭКСПЕРИМЕНТЫ

# ЭКСПЕРИМЕНТЫ: КОДЕКИ

Начиная с версии 19.11 в ClickHouse появились дополнительные кодеки для хранения данных, а именно:

- Gorilla
- DoubleDelta

Первый прекрасно подходит для хранения колонки со значениями (Float64), DoubleDelta же в условиях монотонно возрастающих timestamp, date и version также очень благоприятно влияет на потребляемое место на дисках

- Подробности эксперимента  
в <https://gist.github.com/Felixoid/34b7a37dacf8ff3030ccdb19501e225b>



# ЭКСПЕРИМЕНТЫ: КОДЕКИ

## Оригинальный DDL:

```
CREATE TABLE graphite.data
(
  `metric` String,
  `value` Float64,
  `timestamp` UInt32,
  `date` Date,
  `updated` UInt32
)
ENGINE = GraphiteMergeTree('graphite_rollup')
PARTITION BY toYYYYMMDD(date)
ORDER BY (metric, timestamp)
SETTINGS index_granularity = 8192
```

## Результат эксперимента:

```
CREATE TABLE graphite.data_3days_i256
(
  `metric` LowCardinality(String),
  `value` Float64 CODEC(Gorilla, LZ4),
  `timestamp` UInt32 CODEC(DoubleDelta, LZ4),
  `date` Date CODEC(DoubleDelta, LZ4),
  `updated` UInt32 CODEC(DoubleDelta, LZ4)
)
ENGINE = GraphiteMergeTree('graphite_ig_rollup')
PARTITION BY toYYYYMMDD(
  toStartOfInterval(date, toIntervalDay(3))
)
ORDER BY (metric, timestamp)
SETTINGS index_granularity = 256
```

# ЭКСПЕРИМЕНТЫ: КОДЕКИ

table	name	compressed	uncompressed	ratio	marks	codec
data_3days_i256	date	49.96 MiB	20.38 GiB	417.7	978.38 MiB	CODEC(DoubleDelta, LZ4)
data_3days_original	date	122.92 MiB	20.38 GiB	169.7	30.57 MiB	
data_3days_i256	metric	315.89 MiB	21.39 GiB	69.3	1.91 GiB	
data_3days_original	metric	2.67 GiB	588.30 GiB	220.1	30.57 MiB	
data_3days_i256	timestamp	199.22 MiB	40.77 GiB	209.5	978.38 MiB	CODEC(DoubleDelta, LZ4)
data_3days_original	timestamp	9.33 GiB	40.77 GiB	4.3	30.57 MiB	
data_3days_i256	updated	1.53 GiB	40.77 GiB	26.6	978.38 MiB	CODEC(DoubleDelta, LZ4)
data_3days_original	updated	11.51 GiB	40.77 GiB	3.5	30.57 MiB	
data_3days_i256	value	15.63 GiB	81.53 GiB	5.2	978.38 MiB	CODEC(Gorilla, LZ4)
data_3days_original	value	21.07 GiB	81.53 GiB	3.8	30.57 MiB	

# ЭКСПЕРИМЕНТЫ: groupArray

- И graphouse, и graphite-clickhouse при запросе из ClickHouse выполняют одни и те же `SELECT Path, Time, Value, Timestamp`.
- Изменив запрос на `SELECT Path, groupArray(Time), groupArray(Value), groupArray(Timestamp)`, данные, передаваемые по сети, уменьшились с 1.76 GiB до 285.3 MiB, более чем в 6 раз.
- Патч уже в <https://github.com/lomik/graphite-clickhouse/pull/61>





# С ВАМИ БЫЛ



Михаил Ширяев

System Administrator

System Administration

[mikhail.shiryaev@innogames.com](mailto:mikhail.shiryaev@innogames.com)



InnoGames GmbH

Friesenstrasse 13

20097 Hamburg

<https://www.innogames.com>



ВОПРОСЫ? КОММЕНТАРИИ?

БОЛЬШОЕ СПАСИБО ЗА ВНИМАНИЕ!