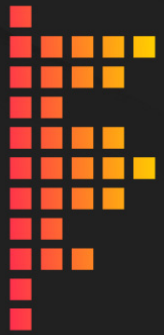


ClickHouse для инженеров и архитекторов БД

Мониторинг и поддержка, диагностика неполадок

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Ставим “+”, если все хорошо
“-”, если есть проблемы

Тема вебинара

Мониторинг и логирование

Цыкунов Алексей

Co-founder & CTO at Hilbert Team

- Более 20 лет опыта в проектировании и реализации отказоустойчивых и высоконагруженных информационных систем в таких отраслях как телеком и FinTech
- Автор курсов по Linux в Otus.ru
- Более 8 лет опыта оптимизации работы продуктовых команд и R&D департаментов с помощью DevOps инструментов и методик (Kubernetes, CI/CD, etc.) и облачных технологий (AWS, GCP, Azure, Yandex.Cloud)

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в учебной группе
#OTUS ClickHouse-2023-11

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

Маршрут вебинара

Цели вебинара

К концу занятия вы сможете

1. настраивать мониторинг ClickHouse
2. настраивать работу с логами в ClickHouse

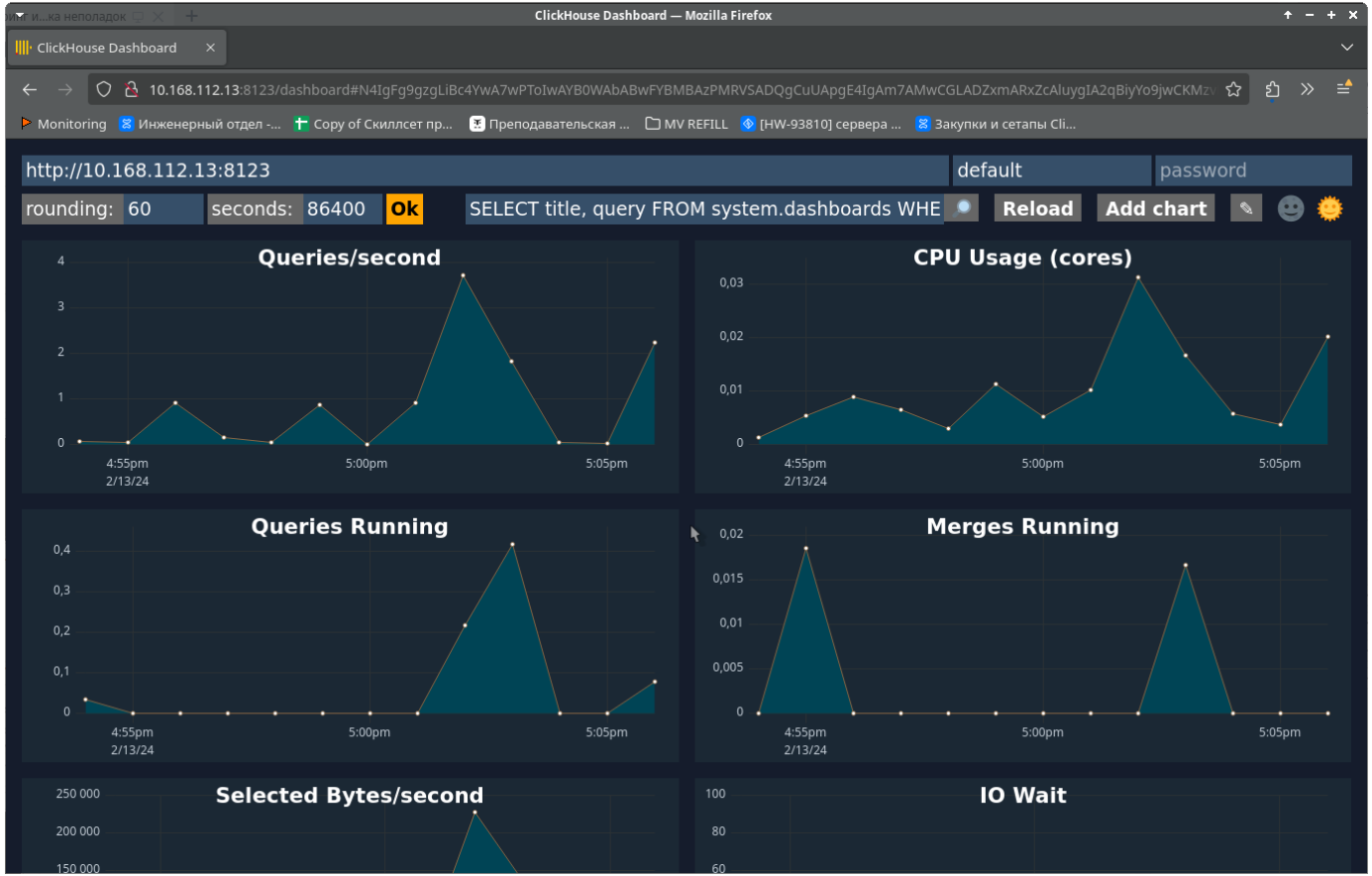
Смысл

Зачем вам это уметь

1. оперативно реагировать на отклонения от штатных показателей работы
2. поддерживать работоспособность системы

Мониторинг

Встроенный дашборд



Встроенный дашборд

Доступен по url http://clickhouse-server:port/dashboard
где clickhouse-server это адрес сервера
а port - http port сервера, по умолчанию 8123

Показывает графики на базе запросов, возвращающих две колонки:
DateTime, любой-числовой-тип

Есть встроенный пресет запросов, в system.dashboards

Можно добавлять свои запросы, но они не сохраняются, а кодируются прямо в URL

Можно создавать свои таблицы с такой же структуры как system.dashboard, и загружать запросы оттуда.

Встроенный дашборд

Доступные метрики:

- Queries/second
- CPU Usage
- Queries Running
- Merges Running
- Selected Bytes/second
- Inserted/Selected Rows/second
- Read From Disk
- Read From FS
- IO Wait
- CPU Wait
- CPU Usage user/system
- Memory usage
- Total parts / Max Parts for Partition
- Можно добавлять свои используя запросы в KX
- Можно сохранять свои в таблицу аналогичной system.dashboards структуры

Graphite & Prometheus

ClickHouse подерживает сбор метрик внешними системами.

Graphite настраивается секцией конфигурации <graphite>
Prometheus настраивается секцией конфигурации <prometheus>

Данные забираются из таблиц:
- system.metrics
- system.events
- system.asynchronous_metrics

Graphite & Prometheus

Пример для Graphite:

```
<graphite>
    <host>localhost</host>
    <port>42000</port>
    <timeout>0.1</timeout>
    <interval>60</interval>
    <root_path>one_min</root_path>
    <metrics>true</metrics>
    <events>true</events>
    <events_cumulative>false</events_cumulative>
    <asynchronous_metrics>true</asynchronous_metrics>
</graphite>
```

Graphite & Prometheus

Пример для Prometheus:

```
<prometheus>
    <endpoint>/metrics</endpoint>
    <port>9363</port>
    <metrics>true</metrics>
    <events>true</events>
    <asynchronous_metrics>true</asynchronous_metrics>
    <errors>true</errors>
</prometheus>
```

Системные таблицы

system.metrics - мгновенные метрики (текущее значение)

metric (String) — Metric name.

value (Int64) — Metric value.

description (String) — Metric description.

system.events - счетчики различных событий

event (String) — Event name.

value (UInt64) — Number of events occurred.

description (String) — Event description.

system.asynchronous_metrics - системные метрики, рассчитываемые в фоне

metric (String) — Metric name.

value (Float64) — Metric value.

description (String - Metric description)

Системные представления

system.processes.

Запросы, выполняемые в настоящий момент, отображаются в представлении

system.mutations

Для отображения мутаций

system.merges

отображает выполняемые в данные момент мутации и слияния таблиц семейства MergeTree.

system.replicated_fetches

показывает выполняемые в данный момент фоновые операции скачивания датапартов с других реплик.

system.replication_queue

содержит информацию о задачах из очередей репликации, хранящихся в ZooKeeper.

system.merges

- метрики отображаются только в реальном времени.
- глобальные метрики слияния фиксируются в system.metrics и system.events.
- тип действия - слияние или мутация определяется флагом **is_mutation**

```
SELECT
    database,
    table,
    elapsed,
    progress,
    rows_read
FROM system.merges
WHERE is_mutation = 1 -- только процессы мутаций датапартов
```

system.errors

- Содержит коды ошибок с указанием количества срабатываний.

```
SELECT
    name,
    value,
    last_error_message
FROM system.errors
ORDER BY value DESC;

SELECT *
FROM system.errors
ORDER BY last_error_time DESC;
```

system.metrics

- Типы метрик
  - *metric ilike '%part%'* — метрики кусков.
  - *metric ilike '%thread%'* — метрики потоков.
  - *metric ilike '%connect%'* — подключения.

- На что обращать внимание
  - Background**PoolTask* < *Background**PoolSize:
    - PoolTask - сколько сейчас активно,
    - PoolSize - размер пула.
  - DeelayedInserts
    - счетчик замедленных операций INSERT ввиду большого количества партов.
  - ContextLockWait
    - блокировки во что угодно, сеть или диск, или просто взаимоблокирующие запросы

system.metrics

```
SELECT
    name,
    value
FROM system.settings
WHERE name ILIKE '%pool%'
```

```
SELECT
    metric,
    value
FROM system.metrics
WHERE metric ILIKE '%pool%'
```

system.events

- Счетчики событий с момента старта системы
  - при перезапуске - сбрасываются

- На что обращать внимание
  - **FailedQuery / FailedInsertQuery / FailedSelectQuery**
    - неуспешные запросы
  - **QueryTimeMicroseconds**
    - счетчик времени затраченного на запросы, можно взять дельту и получится метрика производительности

```
SELECT
    event,
    value
FROM system.events
WHERE event ILIKE '%fail%'
ORDER BY value DESC
```

system. asyncronous_metrics

вычисляемые в фоне метрики

- метрики сетевых адаптеров;
- метрики оперативной памяти;
- метрики дисковой подсистемы;
- метрики процессора.

- *metric ilike '%repl%'* — по репликации.
- *metric ilike '%thread%'* — по потокам.
- *metric ilike '%memo%'* — по памяти.

system.replicated_fetches

Скачивание кусков данных с других реплик

- прогресс выполнения задач;
- количество байтов;
- хост реплики источника и ещё ряд атрибутов.

system.replication_queue

содержит информацию о задачах из очередей репликации, хранящихся в ZooKeeper

Задачи в очереди репликации могут быть как неуспешными, так и **is_currently_executing = 1** — выполняемыми в данный момент.

Можно определить:

- Сколько задач по репликации находится в процессе в данный момент.
- Есть ли задачи, которые не могут выполниться из-за какой-либо ошибки, **is_currently_executing = 0**. В поле **last_exception** будет описание исключения, которое произошло при выполнении задачи, и **num_tries**, показывающее количество неудачных попыток.
- Сколько задач и по какой причине отложено (поле **postpone_reason**).

system.replication_queue

```
SELECT
    table,
    type,
    is_currently_executing,
    count() AS cnt
FROM system.replication_queue
GROUP BY
    table,
    type,
    is_currently_executing
ORDER BY
    is_currently_executing ASC,
    cnt DESC
```

История значений метрик system.metric_log

- содержит историю значений метрик из системных представлений system.metrics и system.events
- к названиям метрик добавляются префиксы **CurrentMetric_*** и **ProfileEvent_*** соответственно
- Для каждой метрики предназначена отдельная колонка
- можно использовать как источник данных для Grafana для построения дашбордов

```
SELECT
    event_time,
    CurrentMetric_HTTPConnection
FROM system.metric_log
ORDER BY CurrentMetric_HTTPConnection DESC
```

Настройка частоты сброс метрик в лог system.metric_log

```
<metric_log>
    <database>system</database>
    <table>metric_log</table>
    <flush_interval_milliseconds>7500</flush_interval_milliseconds>
    <collect_interval_milliseconds>1000</collect_interval_milliseconds>
</metric_log>

<asynchronous_metric_log>
    <database>system</database>
    <table>asynchronous_metric_log</table>
    <flush_interval_milliseconds>7000</flush_interval_milliseconds>
</asynchronous_metric_log>
```

Персонализация мониторинга Prometheus

ClickHouse умеет отдавать результаты заранее заданного запроса, с заданного url, по http-интерфейсу.

А так же, имеет Prometheus-формат для возврата результата запроса.

Этого достаточно, чтобы предоставить метрики в понятном Prometheus-скрейперу формате.

Выдача метрик в Prometheus

Настройка http-handler:

```
<http_handlers>
    <rule>
        <url>/predefined_query</url>
        <methods>POST,GET</methods>
        <handler>
            <type>predefined_query_handler</type>
            <query>SELECT * FROM system.metrics LIMIT 5 FORMAT Template SETTINGS format_template_resultset =
'prometheus_template_output_format_resultset', format_template_row = 'prometheus_template_output_format_row',
format_template_rows_between_delimiter = '\n'</query>
        </handler>
    </rule>
    <rule>...</rule>
    <rule>...</rule>
</http_handlers>
```

Персонализация мониторинга Prometheus

Настройка http-handler:

```
<clickhouse>
    <http_handlers>
        <rule>
            <url>/custom_metrics</url>
            <methods>POST,GET</methods>
            <handler>
                <type>predefined_query_handler</type>
                <query>SELECT * FROM system.custom_prom_metrics FORMAT Prometheus</query>
                <content_type>text/plain; charset=utf-8</content_type>
            </handler>
        </rule>
    </http_handlers>
</clickhouse>
```

Персонализация мониторинга Prometheus

Источник для метрик:

```
CREATE DATABASE custom_prom_metrics;
-- CREATE VIEW custom_prom_metrics.вьюшка AS SELECT запрос;
-- некоторое количество таких VIEW
CREATE TABLE system.custom_prom_metrics;
ENGINE = merge(custom_prom_metrics,”);
```

пример VIEW:

```
CREATE VIEW custom_prom_metrics.merges AS
SELECT
    'merges' AS name,
    count() AS value,
    'active merges' AS help,
    map('hostname', hostName()) AS labels,
    'gauge' AS type
```

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Логирование

Логирование в текстовые файлы

- обычный текстовый лог и текстовый лог для ошибок:

- Для обычного так же задается уровень логирования:
    - trace, debug, information, warning, error, trace

- так же есть встроенная ротация, опциями:
    - count - сколько последних файлов лога хранить
    - size (в байтах) - каким размером файлы

Логирование в текстовые файлы

```
<logger>
    <level>trace</level>
    <log>/var/log/clickhouse-server/clickhouse-server-%F-%T.log</log>
    <errorlog>/var/log/clickhouse-server/clickhouse-server-%F-%T.err.log</
errorlog>
    <size>1000M</size>
    <count>10</count>
    <stream_compress>true</stream_compress>
</logger>
```

Логирование в таблицы

Расширенный функционал логирования доступен посредством записи логов в сам ClickHouse в качестве таблиц. Настраивается одноименной логу секцией конфигурации, например для text_log:

```
<text_log>
    <database>system</database>
    <table>text_log</table>
    <flush_interval_milliseconds>7500</flush_interval_milliseconds>
    <max_size_rows>1048576</max_size_rows>
    <reserved_size_rows>8192</reserved_size_rows>
    <buffer_size_rows_flush_threshold>524288</buffer_size_rows_flush_threshold>
    <flush_on_crash>false</flush_on_crash>
    <level></level>
</text_log>
```

Таблицы логов

- **system.crash_log** содержит информацию о трассировках стека для фатальных ошибок. Создается только при возникновении фатальных ошибок. из этой таблицы можно извлечь следующую информацию:
    - Номер сигнала, пришедшего в поток
    - Идентификатор потока
    - Идентификатор запроса
    - Трассировка стека в момент ошибки
- Если в системе не происходило аварийное завершение потоков, то таблица *crash_log* отсутствует.
- **system.part_log** содержит информацию о всех событиях, произошедших с кусками данных таблиц семейства MergeTree (например, события добавления, удаления или слияния данных).
- **system.query_log** содержит лог запросов.
- **system.query_thread_log** содержит логи потоков, связанных с запросами, которые отдельно логируются в таблицу **system.query_log**.

Таблицы логов

- **system.session_log** содержит логи аутентификации пользователей. По этим логам можно понять, какие учетные записи и с каким типом аутентификации подключались, в том числе ip-адреса хостов с которых выполнялось подключение.

- **system.text_log** содержит логи, записываемые в файлы.

- **system.trace_log** - содержит экземпляры трассировки стека адресов вызова, собранные с помощью семплирующего профайлера запросов

system.query_log

Каждый выполненный запрос будет храниться в истории в виде двух записей с type='QueryStart' и type='QueryFinish'.

- **type** (Enum8) — тип события, произошедшего при выполнении запроса. Значения:
  - **'QueryStart' = 1** — успешное начало выполнения запроса.
  - **'QueryFinish' = 2** — успешное завершение выполнения запроса.
  - **'ExceptionBeforeStart' = 3** — исключение перед началом обработки запроса.
  - **'ExceptionWhileProcessing' = 4** — исключение во время обработки запроса.

system.query_log

```
SELECT
    type,
    event_time,
    query_duration_ms,
    initial_query_id,
    formatReadableSize(memory_usage) AS memory,
    ProfileEvents['UserTimeMicroseconds'] AS userCPU,
    ProfileEvents['SystemTimeMicroseconds'] AS systemCPU
FROM system.query_log
ORDER BY query_duration_ms DESC
LIMIT 10
```

system.part_log

содержит информацию обо всех событиях, произошедших с кусками данных таблиц семейства MergeTree.

Действия над датапартами могут быть следующими:

- NEW_PART - вставка нового датапарта
- MERGE_PARTS - слияние датапартов
- DOWNLOAD_PART - скачивание датапарта с реплики
- REMOVE_PART - удаление датапарта или отсоединение командой DETACH PARTITION
- MUTATE_PART - к датапарту применена мутация
- MOVE_PART - перемещение датапарта с диска на диск

system.query_thread_log

Содержит информацию о потоках, которые порождают запросы.

В таблице **system.query_thread_log** фиксируются имя потока, время его запуска, продолжительность обработки запроса, а также счетчики производительности (в поле **ProfileEvents**).

Все потоки записываются в таблицу **system.query_thread_log**.
В таблице хранится только идентификатор связанного запроса записанного в **system.query_log**.

Если логирование потоков включено на глобальном уровне, то для некоторых запросов можно отключить логирование потов с помощью установки переменной сессии

SET log_query_threads = 0

или определением этой переменной на уровне профиля пользователя.

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Мониторинг проблем

Оценка проблем за последние 5 минут

```
SELECT
        sum(ProfileEvent_FailedQuery),
        sum(ProfileEvent_FailedSelectQuery),
        sum(ProfileEvent_FailedInsertQuery),
        sum(ProfileEvent_ReplicatedPartFailedFetches),
        sum(ProfileEvent_ReplicatedPartChecksFailed),
        sum(ProfileEvent_DistributedConnectionFailTry),
        sum(ProfileEvent_ReplicatedDataLoss)
FROM clusterAllReplicas(default, system.metric_log)
WHERE event_time > now() - interval 5 minute
```

Прерванные мутации за последние 24 часа

```
SELECT table, mutation_id, command, latest_fail_reason
FROM cluster(default, system.mutations)
WHERE
        is_done = 0 AND
        latest_fail_time > now() - interval 24 hour
```

Ошибки репликации за последний час

```
SELECT table, replica_name, node_name, merge_type
FROM cluster(default, system.replication_queue)
WHERE
        last_exception is not null and
        last_attempt_time > now() - interval 1 hour
```

Выявление долгих запросов

```
SELECT
        _shard_num,
        query_start_time,
        query_duration_ms,
        query,
        type,
        read_rows,
        read_bytes,
        memory_usage
FROM clusterAllReplicas(default, system.query_log)
WHERE
        exception_code = 0 and
        query_start_time > now() - interval 7 day and
        type = 'QueryFinish'
ORDER BY query_duration_ms desc
LIMIT 10\G
```

Домашнее задание

Вариант 1

1) Придумать 2 или более запросов для персонализированного мониторинга.
2) Создать таблицу с этими запросами в нужном формате
3) Предоставить скриншот встроенного дашборда, показывающего эти таблицы

Вариант 2

1) Развернуть Graphite/Prometheus.
2) Настроить push/pull метрик
3) Предоставить скриншот результата сбора любой метрики

По желанию, дополнительно

1) Настроить для любой таблицы с логами Engine=Null
2) Создать рядом дополнительную реплицируемую таблицу такой же структуры, но с любым материализованным полем, идентифицирующим реплику.
3) Настроить MV для захвата логов из Engine=Null таблицы в реплицируемую.
4) Поднять +1 реплику, убедиться что логирование теперь реплицируемо.
5) Предоставить секции конфигурации и «CREATE TABLE» таблиц для проверки.

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Рефлексия

Вопросы для проверки

1. какие метрики мониторить в первую очередь
2. как исправить ошибку Too Many Parts

Рефлексия

С какими впечатлениями уходите с вебинара?

Как будете применять на практике то,
что узнали на вебинаре?

Заполните, пожалуйста, опрос о занятии по ссылке в чате