

РОССИЯ СЕГОДНЯ

Особенности мониторинга и траблшутинга высоконагруженной БД PostgreSQL

pgconf 2018
5-7 февраля 2018



#PostgreSQLRussia

Дмитрий Кремер
Администратор баз данных
E-mail: d.kremer@rian.ru
dmitry.kremer@gmail.com

МИА «Россия сегодня»

- Ведущее международное новостное агентство с 1941 года (тогда СовИнформБюро)
- Крупнейший поставщик новостного и медиа-контента в Российской Федерации (бренды РИА Новости и Sputnik News)
- Фотохостинг Олимпиады в Сочи 2014
- Десятки корреспондентов по всей России
- Современные мультимедиа-прессцентры в Москве и Симферополе
- Платформы в социальных сетях
- Производство и распространение фотоконтента, инфографики, контента для мобильных приложений.

Дмитрий Кремер

- Опыт работы с различными базами данных в качестве разработчика и системного администратора с 1999 года.
- Непрерывный опыт работы с БД Oracle с 2007 года
- Oracle Certified Professional 9i, 10g
- Начал работать с PostgreSQL в мае 2015 года.

Три кита мониторинга

- Сбор метрик
- Алертинг (оповещение)
- Анализ и визуализация собранных метрик

Влияние наблюдателя на наблюдаемую систему

- Двухщелевой эксперимент
- Квантовые эффекты – что имеется ввиду
- При чём тут БД ?

Какие метрики нужно собирать

Система

- CPU (user/system/iowait/idle), диски, память, swap
- Значения настроек ядра/ОС и их изменения
- Время отклика по сети

PostgreSQL

- Наличие специфичных для PostgreSQL процессов в ОС
- Время отклика на запрос select 1;
- Количество сессий во всевозможных разрезах (active/idle/idle in transaction, waiting, базы, пользователи и т. д.)
- Значения критически важных параметров (fsync, synchronous_commit и т. д.)
- WAL, репликация, бэкап
- Статистические данные pg_stat_*
- Блокировки

Как собирать метрики

- Стремиться максимально сериализовать сбор метрики
- Если значение метрики – результат выполнения запроса к БД, стремиться получить максимальное количество метрик в одном запросе
- Разделить сбор и хранение метрик

Zabbix – yet another monitoring system

- Почему именно Zabbix ?

Сильные стороны

- Шаблоны (Templates)
- Динамические правила создания объектов мониторинга (Discovery Rules)
- Расширяемая архитектура

Слабые стороны

- Монструозный комбайн
- Host – ориентированный мониторинг

Инструменты анализа проблем

- Анализ текущей нагрузки на сервер: sar, top (его вариации), iotop, iostat, netstat
- Анализ сессий: pg_stat_activity, статистика pgbouncer
- Анализ выполняющихся запросов: pg_stat_activity, pg_stat_statements
- Анализ логов: pgBadger, скрипты на awk
- Анализ графиков Zabbix и/или других имеющихся средств визуализации собранных метрик и логов

Локализация проблемы

- Во всем виновата база данных (придумайте свою вариацию) (???)
- Время отклика – основной маркер проблемы
- Высокий CPU system = конкуренция за ресурсы (в 95% случаев за память)
- Популярные причины высокого CPU iowait – большое количество сортировок на диске, низкая селективность запросов
- Причина большого времени отклика БД – неоптимальное выполнение SQL-запросов

Источники для анализ запросов

- `pg_stat_activity`

Особенность: только моментальный срез

- `pg_stat_statements`

Плюсы: Все запросы за определённый период

Минусы: Нет хронологии, нет планов запросов

- логи БД

Плюсы: Есть все запросы, есть планы долгих запросов (`auto_explain` extention)

Минусы: Огромный объём, плохо подходит для оперативного анализа

Архитектура системы сохранения запросов



ClickHouse



Задачи при проектировании системы

- Сохранить формат лога доступным для обработки pgBadger
- Иметь возможность «скормить» имеющиеся логи системе хранения
- Преобразовать данные лога PostgreSQL в JSON для передачи их в Clickhouse через Kafka, собрать из потока многострочного лога единую запись с полным текстом SQL-запроса
- Минимизировать затраты на программирование. По возможности использовать существующие поддерживаемые решения.

Варианты парсинга логов в JSON

- С помощью хука PostgreSQL средствами модифицированного подгружаемого модуля jsonlog от Michael Paquier (Мишель Пакер) <http://paquier.xyz/>

Ломается формат лога, который способен обработать pgBadger

- Читать поток от PostgreSQL в формате syslog из rsyslog с помощью модуля imtcp применяя правила постобработки. Далее формировать JSON на уровне шаблона.

Не получилось с помощью параметров модуля реализовать склейку строк

- Читать лог PostgreSQL в формате stderr из rsyslog с помощью модуля imfile применяя правила постобработки. Далее формировать JSON на уровне шаблона.

Склейка строк получилась. Также с помощью модуля imfile можно загрузить в систему имеющиеся логи.

Конфигурация PostgreSQL

log_destination = 'stderr'

log_duration = on

logging_collector = on

log_lock_waits = on

log_autovacuum_min_duration = 0

log_directory = '/data/log/pgsql'

log_filename = 'postgresql-serverlog-%a.log'

log_rotation_age = 1440

log_truncate_on_rotation = on

log_file_mode = 0644

log_rotation_size = 0

log_min_messages = log

log_min_duration_statement = 0ms

log_checkpoints = on

log_connections = on

log_disconnections = on

log_line_prefix = '%t [%p]: [%l] user=%u,db=%d '

Конфигурация rsyslog-kafka

```
module(load="omkafka") # provides kafka support
main_queue(
  queue.workerthreads="3" # threads to work on the queue
  queue.dequeueBatchSize="200" # max number of messages to
  process at once
  queue.size="22000" # max queue size
)
template(name="json_lines" type="list" option.json="on") {
  constant(value="{")
  constant(value="\ttimestamp\":"")
  property(name="timereported" dateFormat="year")
  constant(value="-")
  property(name="timereported" dateFormat="month")
  constant(value="-")
  property(name="timereported" dateFormat="day")
  constant(value=" ")
  property(name="timereported" dateFormat="hour")
  constant(value=":")
  property(name="timereported" dateFormat="minute")
  constant(value=":")
  property(name="timereported" dateFormat="second")
```

```
constant(value="\,\"duration\":"")
property(name="rawmsg"
  regex.type="ERE"
  regex.submatch="1"
  regex.expression="duration: ((([0-9]+).([0-9]+))"
  regex.nomatchmode="BLANK"
)
constant(value="\,\"user\":"")
property(name="rawmsg"
  regex.type="ERE"
  regex.submatch="1"
  regex.expression="user=([a-z]+),"
  regex.nomatchmode="BLANK"
)
constant(value="\,\"db\":"")
property(name="rawmsg"
  regex.type="ERE"
  regex.submatch="1"
  regex.expression=",db=([a-z]+)"
  regex.nomatchmode="BLANK"
)
```

Конфигурация rsyslog-kafka (продолжение)

```
constant(value="\\,\\statement\\:\\")
property(name="rawmsg"
  regex.type="ERE"
  regex.submatch="1"
  regex.expression="statement: (.+)"
  regex.nomatchmode="BLANK"
)
constant(value="\\,\\host\\:\\")
property(name="hostname")
constant(value="\\,\\severity\\:\\")
property(name="syslogseverity-text")
constant(value="\\,\\facility\\:\\")
property(name="syslogfacility-text")
constant(value="\\,\\syslog-tag\\:\\")
property(name="syslogtag")
constant(value="\\}")
}
```

```
ruleset(name="sendToKafka") {
  if $msg contains 'duration:' then
    action(broker=["localhost:9092"]
      type="omkafka" topic="syslog_pg"
      template="json_lines"
      confParam=["socket.timeout.ms=1000",
        "socket.keepalive.enable=true"])
}

input(type="imfile"
  File="/data/log/pgsql/*.log"
  tag="postgres"
  startmsg.regex="^# Time: [0-9]{6}"
  readTimeout="2"
  escapeLF="off"
  Ruleset="sendToKafka")
```


Пример полученного JSON

```
{ "timestamp": "2018-01-12 15:18:12", "duration": "1.960", "user": "postgres", "db": "postgres", "statement": "SELECT d.datname as \"Name\",  
pg_catalog.pg_get_userbyid(d.datdba) as \"Owner\",  
pg_catalog.pg_encoding_to_char(d.encoding) as \"Encoding\",  
d.datcollate as \"Collate\",  
d.datctype as \"Ctype\",  
pg_catalog.array_to_string(d.datacl, E'\\n') AS \"Access privileges\",  
CASE WHEN pg_catalog.has_database_privilege(d.datname, 'CONNECT')  
THEN pg_catalog.pg_size_pretty(pg_catalog.pg_database_size(d.datname))  
ELSE 'No Access'  
END as \"Size\",  
t.spcname as \"Tablespace\",  
pg_catalog.shobj_description(d.oid, 'pg_database') as \"Description\"  
FROM pg_catalog.pg_database d  
JOIN pg_catalog.pg_tablespace t on d.dattablespace = t.oid  
ORDER BY 1;"; "host": "pgsql-rhel7-test3" }
```

Конфигурация Kafka-Clickhouse

```
CREATE TABLE queue
(
  timestamp DateTime,
  duration Float32,
  user String,
  db String,
  statement String,
  host String
)
ENGINE = Kafka('localhost:9092','syslog_pg','ch_consumer','JSONEachRow')
```

Резюме

- Необходимо помнить о нагрузке на БД со стороны мониторинга и стремиться свести её к минимуму
- Время отклика – основной показатель наличия проблем с БД
- Основной вклад в решение проблем с БД вносит анализ и оптимизация запросов и/или архитектуры приложения

Полезные ссылки

- <https://habrahabr.ru/post/321262/> - статья про сбор логов с rsyslog
- <http://www.rsyslog.com/doc/v8-stable/configuration/templates.html> - документация по шаблонам rsyslog
- <http://www.rsyslog.com/regex/> - тестирование регехр для rsyslog
- <http://www.rsyslog.com/rainerscript-constant-string-escaper/> - экранирование регехр для конфигурационного файла
- https://clickhouse.yandex/docs/en/table_engines/kafka.html - адаптер Kafka для Clickhouse
- <https://www.artofmonitoring.com/> - Книга Art of Monitoring

Спасибо за внимание!

Вопросы ?

dmitry.kremer@gmail.com

d.kremer@rian.ru

РОССИЯ  **СЕГОДНЯ**