Arenadata™ Database

Версия - v6.8.1_arenadata5

Коннектор ADB и Clickhouse

2020 Arenadata™

Оглавление

1 Архитектура коннектора **3**

2 Установка **4**
2.1 Установка с помощью ADCM 4
2.2 Установка из rpm-пакетов 4
2.3 Установка расширения (gptkh) 5

3 Конфигурация коннектора и эксплуатация **6**
3.1 Системные требования 6
3.2 Модуль PXF 6
3.3 Расширение 8
3.4 Очистка промежуточных таблиц 9

4 Пример интеграции **10**
4.1 Без использования расширения 10
4.2 С использованием расширения 10

В документе приведено описание коннектора Arenadata DB с Clickhouse.

Important: Контактная информация службы поддержки – e-mail: info@arenadata.io

Глава 1

Архитектура коннектора

Arenadata DB (**ADB**) это горизонтально-масштабируемая СУБД, предназначенная для построения больших аналитических хранилищ данных (от источников до витрин данных). Такие модели требуют наличия множества соединений, развитый синтаксис ANSI SQL, транзакционность и другие возможности. С другой стороны **Clickhouse** предоставляет доступ к построенным широким витринам с максимальной скоростью.

Коннектор поддерживает только возможность односторонней вставки данных. Он состоит из двух компонентов – модуля **PXF**, непосредственно занимающегося отправкой данных с сегментов, и расширения **ADB**, предназначенного для того, чтобы сделать операцию загрузки данных более безопасной при отсутствии транзакций в **Clickhouse**.

Модуль **PXF** использует функциональность внешних таблиц **ADB** и может быть использован отдельно. Он не делает различия в типе таблиц **Clickhouse** и отправляет данные равномерно на те ноды, которые указаны в его конфигурации. Модуль содержит необходимую логику для регулирования скорости отправки данных, которую требует движок слияния таблиц **Clickhouse**. Также архитектура модуля позволяет задавать дополнительные настройки, влияющие на производительность, такие как количество сетевых потоков, длину очереди и прочие.

Расширение **ADB** напротив, учитывает топологию таблиц **Clickhouse** и создает в точности ее повторяющий промежуточный слой. Это необходимо как для обеспечения консистентности, так и для оптимальной производительности. Как только данные загружаются в промежуточный слой, расширение проверяет совпадение по количеству строк на физических таблицах и переключает куски данных из промежуточного слоя в реальные таблицы. Эта операция не требует перемещения данных. Расширение для пользователя представляет собой процедуру, в которую следует полностью передать запрос `INSERT` во внешнюю таблицу.

Глава 2

Установка

Функциональность коннектора включена только в Enterprise версию бандла ADB.

2.1 Установка с помощью ADCM

Для установки коннектора с помощью **ADCM** требуется инсталлировать сервис *PXF* на все сегментные ноды кластера, а также в списке сервисов выбрать сервис *Tkhemali* – кодовое имя коннектора. При этом необходимые пакеты и файлы автоматически устанавливаются на машины кластера.

2.2 Установка из rpm-пакетов

Инструкция из rpm-пакетов предполагает, что сервис *PXF* запущен на сегментных нодах и работает.

Установка модуля *PXF* (*tkh-connector*):

1. Установить пакет *tkh-connector* на тех нодах, где запущен сам сервис *PXF*.
2. Добавить в файл профилей */etc/pxf/conf/pxf-profiles-default.xml* на каждой ноде следующую секцию:

```
<profiles>
    <profile>
        <name>Tkh</name>
        <description>Clickhouse</description>
        <plugins>
            <accessor>io.arenadata.tkh.TkhAccessor</accessor>
            <resolver>io.arenadata.tkh.TkhResolver</resolver>
        </plugins>
        <optionMappings>
            <mapping option="req_length_max" property="clickhouse.request.length.max"/>
            <mapping option="req_volume_max" property="clickhouse.request.volume.max"/>
            <mapping option="send_threads" property="clickhouse.send.threads"/>
            <mapping option="send_delay" property="clickhouse.send.delay"/>
            <mapping option="send_queue_sizeMultiplier" property="clickhouse.send.queue.sizeMultiplier"/>
            <mapping option="net_timeout" property="clickhouse.network.timeout"/>
        </optionMappings>
    </profile>
</profiles>
```

3. Перезапустить сервис *PXF* на всех нодах кластера.

2.3 Установка расширения (gptkh)

Для установки расширения (*gptkh*) необходимо:

1. Установить пакет *gptkh* на всех нодах кластера.
2. Загрузить расширение и поддержку языка *python* на мастере:

```
create extension plpythonu;
create extension gptkh;
```

Important: В ADB 5.x поддержка python включается по команде `create language plpythonu;`

Глава 3

Конфигурация коннектора и эксплуатация

3.1 Системные требования

Для работы коннектора необходимо, чтобы все ноды кластера **ADB** имели доступ к TCP-порту *8123* на всех нодах кластера **Clickhouse**, через которые планируется производиться загрузка данных.

3.2 Модуль PXF

Для загрузки данных в **Clickhouse** необходимо создать внешнюю таблицу, указав в ее директиве *LOCATION* протокол *PXF* с профилем модуля *TKH* и его параметры.

3.2.1 Синтаксис

```
CREATE WRITABLE EXTERNAL TABLE <table_name> (
    { <column_name> <data_type> [, ...] | LIKE <other_table> }
)
LOCATION (
    'pxf://<clickhouse_table_name>?<pxf_parameters><settings>'
)
FORMAT 'TEXT'
ENCODING 'UTF8';
```

Где `<pxf_parameters>`:

```
{ PROFILE=TKH | ACCESSOR=io.arenadata.tkh.TkhAccessor&RESOLVER=io.arenadata.tkh.TkhResolver }
[ &SERVER=<server_name> ]
```

Значение `<server_name>` и `<settings>` описаны далее.

3.2.2 Параметры модуля

- **URL** – используется для указания нод, на которые равномерно производится загрузка данных.
  - URL ноды кластера Clickhouse или список нод, разделеных запятой `,`
  - Clickhouse URL имеет синтаксис: `<user>:<password>@<host>:<port>`
  - Option: `URL`
  - Параметр конфигурации: `clickhouse.dist.simple.url`

  - Значение: String

- **Максимальная длина запроса (в строках)** – когда длина достигнута, пакет формируется и кладется в очередь на отправку.
  - Option: `req_length_max`
  - Параметр конфигурации: `clickhouse.request.length.max`
  - Значение: Integer >= 0
  - Значение по умолчанию: *76800*
- **Максимальный размер запроса (в байтах)** – когда размер достигнут, пакет формируется и кладется в очередь на отправку.
  - Option: `req_volume_max`
  - Параметр конфигурации: `clickhouse.request.volume.max`
  - Значение: Integer >= 0
  - Значение по умолчанию: *52428800* (50Mib)
- **Число потоков для отправки** – число потоков создается каждым экземпляром *PXF* (число экземпляров *PXF* в данный момент равно числу нод в кластере **ADB**).
  - Option: `send_threads`
  - Параметр конфигурации: `clickhouse.send.threads`
  - Значение: Integer >= 0
  - Значение по умолчанию: *2*
- **Задержка между отправками в потоках** – базовая задержка между двумя последовательными запросами к **Clickhouse**.
  - Option: `send_delay`
  - Параметр конфигурации: `clickhouse.send.delay`
  - Значение: Integer >= 0
  - Значение по умолчанию: *300*
- **Множитель размера очереди для потоков** – модель использует очередь для отправки запросов в отправляющие потоки. Максимальная длина очереди определяется как произведение двух переменных: *число потоков для отправки* и *множитель размера очереди*. Важно обратить внимание, что реальный размер очереди может превышать значение полученного произведения, однако, он никогда не превышает значения: *число потоков для отправки* умноженное на *множитель размера очереди + 1*.
  - Option: `send_queue_sizeMultiplier`
  - Параметр конфигурации: `clickhouse.send.queue.sizeMultiplier`
  - Значение: Double >= 0.0
  - Значение по умолчанию: *2.0*
- **Время ожидание сети (мс)** – время ожидания HTTP-запроса.
  - Option: `net_timeout`
  - Параметр конфигурации: `clickhouse.network.timeout`
  - Значение: Integer >= 0
  - Значение по умолчанию: *60000*

3.2.3 Дополнительно

Для оптимальной производительности стоит указывать распределение внешней таблицы аналогично таблице-источнику, что позволит не перераспределять данные перед отправкой, а слать напрямую с сегментов. Этого можно достигнуть, создавая таблицу через `LIKE source_table` или явно указывая ту же самую колонку в директиве *DISTRIBUTED BY*.

3.3 Расширение

Расширение используется в случаях, когда необходимы дополнительные гарантии консистентности вставки. При этом необходимые настройки кластера **Clickhouse** автоматически проверяются при попытке вставки, используя процедуру расширения, но в частности требуется включенная опция `insert_distributed_sync` для распределенных таблиц.

3.3.1 Создание внешней таблицы

При создании внешней таблицы следует указывать имя промежуточной таблицы, которое зависит от id транзакции. Например, если в **Clickhouse** имя таблицы `default.t`, то в директиве *LOCATION* надо указать `default.t_tmp_$`. Также следует указывать полное имя таблицы вместе с именем базы данных.

Important: В параметрах внешней таблицы нельзя использовать директиву *SERVER*, хранящую название конфигурации PXF в локальных файлах (при этом не указывая параметр *URL*). Расширению требуются адреса хостов кластера Clickhouse, которые оно берет из параметра *URL*

Полное выражение для создания внешней таблицы выглядит следующим образом:

```
create writable external table ext_tct(a int)
location ('pxf://default.d_tct_tmp_$?profile=tkh&url=default@sdch1:8123')
format 'text' encoding 'utf8';
```

Где `sdch1` – имя ноды кластера **Clickhouse**, на которой создана распределенная таблица `default.d_tct`.

3.3.2 Процедура и ее параметры

После создания внешней таблицы в нее можно выполнить вставку с помощью процедуры расширения:

```
select txn('insert into ext_tct select generate_series(1, 10000);');
```

Процедура определена следующим образом:

```
function txn(query text, http_port int default 8123, debug boolean default false, ending_pattern text␣
↪default '_tmp_$')
returns void
```

Параметры процедуры:

- `query` – запрос ADB, который вставляет данные во внешнюю таблицу. Запрос валидируется и разбирается для получения информации о кластере Clickhouse и его таблицах. Удобно использовать `$$ insert into...$$` вместо кавычек (поскольку это позволяет использовать перевод строки и кавычки внутри `$$`);
- `http_port` – общий для всех нод, вовлеченных в данный запрос, HTTP-порт ClickHouse. В данный момент нет возможности получить это значение через SQL (стоит указывать, если отличен от *8123*);
- `debug` – включает логирование в процедуре *txn*. Выводит информацию о всех внутренних операциях: создание промежуточных таблиц, валидация данных, переключение и удаление кусков данных;

- `ending_pattern` – по умолчанию процедура *txn* создает промежуточный слой с очень близкой структурой к оригинальной топологии, но имена таблиц при этом имеют окончания, включающие номер транзакции. Этот параметр позволяет изменить шаблон для временных таблиц.

3.4 Очистка промежуточных таблиц

В случае обрыва транзакции может понадобиться очистка промежуточных таблиц на всех нодах кластера. Для этого можно использовать вспомогательные процедуры и запросы:

1. Определение промежуточных таблиц:

```
select database || '.' || name as tbl from sys_tables('sdch1', 8123)
where database = 'default' and name like 'd_tct_tmp_%';
           tbl
-------------------------
 default.d_tct_tmp_34395
(1 row)
```

2. Теперь, зная, что промежуточные таблицы созданы с номером транзакции *34395*, можно удалить их на всех нодах кластера:

```
with ch(h, p) as (values ('sdch1', 8123), ('sdch2', null))
select drop_staging_tbl_based_on('default', 'd_tct', ch.h, ch.p, 'default', '', false, '_tmp_34395') from␣
↪ch;
```

Глава 4

Пример интеграции

4.1 Без использования расширения

1. Создать таблицу для тестовых данных:

```
create table test (
id int,
text1 text,
text2 text,
) with (appendonly=true, orientation=column, compresstype=zstd, compresslevel=1)
distributed by (id);
```

2. Сгенерировать тестовые данные:

```
insert into test select gen, 'Some text #' || gen::text, 'Some another text #' || gen::text from generate_
↪series(1, 1000000) gen;
```

3. Создать внешнюю таблицу:

```
create writable external table test_ext (
        like table
 )
LOCATION ('pxf://tkemali_test_ext? PROFILE=TKH&URL=sdch1:8123,sdch2:8123&send_threads=4&send_delay=100&send_
↪queue_sizeMultiplier=3.0&req_length_max=5000')
FORMAT 'TEXT'
ENCODING 'UTF8';
```

4. Создать аналогичную таблицу в Clickhouse (шаг выполняется на стороне Clickhouse):

```
CREATE TABLE default.tkhemali_test_ext (`id` Int32, `text1` String, `text2` String) ENGINE = MergeTree␣
↪PARTITION BY tuple() ORDER BY id SETTINGS index_granularity = 8192
```

5. Загрузить данные:

```
INSERT INTO test_ext SELECT * FROM test;
```

4.2 С использованием расширения

Используя таблицы *test* в **ADB** и *tkhemali_test_ext* в **Clickhouse**, созданные ранее, загрузить данные через безопасные процедуры расширения можно следующим образом:

1. Создать внешнюю таблицу, которая указывает на промежуточный слой в Clickhouse:

```
create writable external table test_ext2 (
        like table
 )
LOCATION ('pxf://tkemali_test_ext_tmp_$?PROFILE=TKH&URL=sdch1:8123,sdch2:8123&send_threads=4&send_delay=100&
↪send_queue_sizeMultiplier=3.0&req_length_max=5000')
FORMAT 'TEXT'
ENCODING 'UTF8';
```

2. Загрузить данные, используя процедуру расширения *txn*:

```
select txn('INSERT INTO test_ext2 SELECT * FROM test;');
```