УДК 004.65

Афанасьев Г.И.
кандидат технических наук
доцент кафедры «Системы обработки информации и управления»
МГТУ им. Н.Э.Баумана
Россия, г. Москва
Белоногов И.Б.
старший преподаватель кафедры «Системы обработки информации и управления»
МГТУ им. Н.Э.Баумана
Россия, г. Москва
Булатова И.Г.
доцент кафедры «Системы обработки информации и управления»
МГТУ им. Н.Э.Баумана
Россия, г. Москва
Тоноян С.А.
кандидат технических наук
доцент кафедры «Системы обработки информации и управления»
МГТУ им. Н.Э.Баумана
Россия, г. Москва

ОРГАНИЗАЦИЯ КЛАСТЕРОВ ДЛЯ ОБРАБОТКИ ДАННЫХ НА ОСНОВЕ СУБД YANDEX CLICKHOUSE И РАСПРЕДЕЛЕННОЙ КООРДИНАЦИОННОЙ СЛУЖБЫ ДЛЯ РАСПРЕДЕЛЕННЫХ ПРИЛОЖЕНИЙ APACHE ZOOKEEPER

Аннотация: *В данной статье рассматриваются вопросы по организации кластеров для обработки данных на основе системы управления базами данных YANDEX CLICKHOUSE и распределенной координационной службы для распределенных приложений APACHE ZOOKEEPER*

Ключевые слова*: СУБД, ClickHouse, OLAP, базы данных, ZOOKEEPER, кластер*

Abstract: *This article discusses the organization of clusters for data processing based on the database management system data YANDEX CLICKHOUSE and the distributed coordination service for distributed applications APACHE ZOOKEEPER.*

Keywords: *DBMS, PostgreSQL, ClickHouse, OLAP, databases, ZOOKEEPER, cluster.*

Введение

В настоящее время широкое распространение получили реляционные системы управления базами данных (СУБД), одним из основных логических элементов которых являются таблицы. Как известно, таблицу можно рассматривать как множество либо строковых данных, или также, как и множество столбцовых данных. Наиболее широкое распространение в настоящее время имеют так называемые «строковые» СУБД, одним из популярных представителей которых является СУБД PostgreSQL. Однако в ряде случаев при обработке данных, например для OLAP (online analytical processing) обработки, наиболее целесообразным способом представления и хранения данных является представление данных в виде столбцовых данных. Одной из набирающих популярность СУБД для организации OLAP обработки данных является появившееся не так давно на рынке СУБД ClickHouse от российской компании Yandex. На основе этой СУБД в настоящее временя уже работают такие сервисы как Yandex Метрика, Yandex Маркет и другие крупные проекты Yandex компании. В объединение с другим, открытым проектом Apache Zookeeper **(**распределенной координационной службы для распределенных приложений), возникает возможность организации высокопроизводительного распределенного кластера.

Apache Zookeper и СУБД Yandex ClickHouse

Yandex ClickHouse это система управления базами данных (СУБД), которая для аналитической обработки в реальном времени (англ. online analytical processing OLAP) использует представление и хранение данных ввиде столбцов, а не строк, как обычные СУБД.

На данный момент существует много других решений для работы с большими объемами данными. Приведенная далее классификация этих решений показывает так же преимущества Yabdex ClickHouse относительно этих решений.

1. Коммерческие OLAP СУБД

Примеры: Sybase IQ, HP Vertica, Actian Matrix, EXASol, Actian Vector, и другие.

Отличия Yandex ClickHouse: открытая и бесплатная .

2. Облачные технологии

Примеры: Amazon Redshift и Google BigQuery.

Отличия Yandex ClickHouse: возможность использовать ClickHouse в своей инфраструктуре

3. Надстройки над Hadoop.

Примеры: Cloudera Impala, Spark SQL, Facebook Presto, Apache Drill.

Отличия Yandex ClickHouse:

- ClickHouse позволяет обслуживать аналитические запросы в рамках массового сервиса, такого как Яндекс Метрика;
- для функционирования ClickHouse не требуется разворачивать Hadoop инфраструктуру, этот проект прост в применении, и подходит даже для небольших проектов;
- ClickHouse позволяет загружать данные в реальном времени и самостоятельно занимается их хранением и индексацией;
- в отличие от Hadoop, ClickHouse может работать в территориально распределённых центрах обработки данных.

4. OLAP СУБД с открытм кодом

Примеры: InfiniDB, MonetDB, LucidDB.

Разработка всех этих проектов не доведена до конца и по сути дальнейшие разработки имеют слабую перспективу.

5. Open-source системы для аналитики, не являющиеся Relational OLAP СУБД.

Примеры: Metamarkets Druid, Apache Kylin.

Отличия Yandex ClickHouse: эта СУБД не требует предагрегации данных. ClickHouse поддерживает язык запросов SQL и обеспечивает все сервисы реляционных СУБД.

В рамках той достаточно узкой области, в которой находится ClickHouse, у этой СУБД пока нет достойных альтернатив. Для более широкого применения, ClickHouse может оказаться предпочтительнее других систем с точки зрения скорости обработки запросов, эффективности использования ресурсов и простоты эксплуатации.

Главные особенности СУБД ClickHouse:

- Все столбцы хранятся в отдельных файлах, что позволяет считывать только те столбцы, по которым построен аналитический запрос. В случае, когда в таблице больше 100 столбцов, а запрос строится по 5 из них, то время операций вывода может уменьшаться в 20 раз.
- СУБД использует векторный движок, позволяющий производить обработку данных фрагментов столбцов. ClickHouse строго использует значения постоянной длины, чтобы очистить данные от мусора в виде специальных символов. Что приводит к уменьшению нагрузки на CPU и повышает пропускную способность.
- СУБД ClickHouse является многопоточной СУБД, обеспечивая параллельную обработку запросов.
- Первичный ключ обязателен и должен является датой. Каждый пулл данных с одинаковым первичным ключом ClickHouse хранит в одном представлении.

- СУБД ClickHouse дает возможность репликации и шардирования данных.

Для проведения репликации (синхронизации содержимого нескольких копий данных) требуется Apache ZooKeeper. ClickHouse будет самостоятельно обеспечивать консистентность данных на репликах и производить восстановление после сбоев [1].

Apache ZooKeeper это распределенная координационная служба для распределенных приложений. Или более конкретно, то это распределенное хранилище key/value со следующими основными свойствами:

- распределение ключей представляется в виде дерева (иерархию, подобную файловой системе);
- значения могут содержаться в любом узле иерархического дерева, а не только в листьях дерева
- между клиентом и сервером существует двунаправленная связь, поэтому клиент может подписываться как изменение конкретного значения или части иерархии;
- возможно создать временную пару ключ/значение, которая существует, пока клиент, её создавший, подключен к кластеру;
- все данные должны помещаться в память;
- устойчивость к неработоспособности некритического количества узлов кластера.

Apache ZooKeeper – сервис распределенных приложений, который используется кластером (группой узлов) для согласования друг с другом и поддержания общих данных методами синхронизации. ZooKeeper сам по себе является распределенным приложением, предоставляющим услуги для описания распределенных приложений.

Основные сервисы, предоставляемые ZooKeeper:

- Служба имен - идентификация узлов в кластере по имени. Очень похоже на DNS, но только для узлов

- Управление конфигурацией - последняя и актуальная информация о системе новому подключаемому узлу.
- Управление кластером - добавление/удаление узла виз кластера и статус узла в реальном времени.
- Выбор лидера - выбор узла в качестве лидера в целях согласования.
- Служба блокировки и синхронизации - блокировка данных в момент их изменения. Механизм помогает пользователю в автоматическом восстановлении для случаев возникновения ошибки при подключении других распределенных приложений, таких как Apache HBase.
- Высоконадежная регистрация данных - доступ к данным даже в случае, когда один или несколько узлов стали недоступны или вышли из строя [2].

Преимущества от использования Apache ZooKeeper:

- Простота организации процесса распределенной координации.
- Синхронизация между процессами, то есть взаимное исключение и сотрудничество между серверными процессами. Этот процесс помогает в Apache HBase для управления конфигурациями.
- Упорядоченные сообщения.
- Сериализация - кодирование данных в соответствии с определенными правилами. Необходимо убедиться, что приложение работает стабильно. Этот подход может быть использован в MapReduce для координации очереди для выполнения запущенных потоков.
- Достаточная степень надежности.
- Транзакции осуществляются до конца, а не реализуются частично [3].

Организация кластера

Для реализации минимального возможного надежного хранилища с высокой степенью доступности и скоростью обработки, необходимо порядка 5 серверов. Данные будут разделены на 2 части, и каждая часть будет храниться в двух экземплярах.

Минимальные технические и программные требования к серверам: 512MB RAM, 1CPU, 2GB SWAP, Ubuntu 14.04 x64 OS.

На рисунке 1 представлена топология кластера.

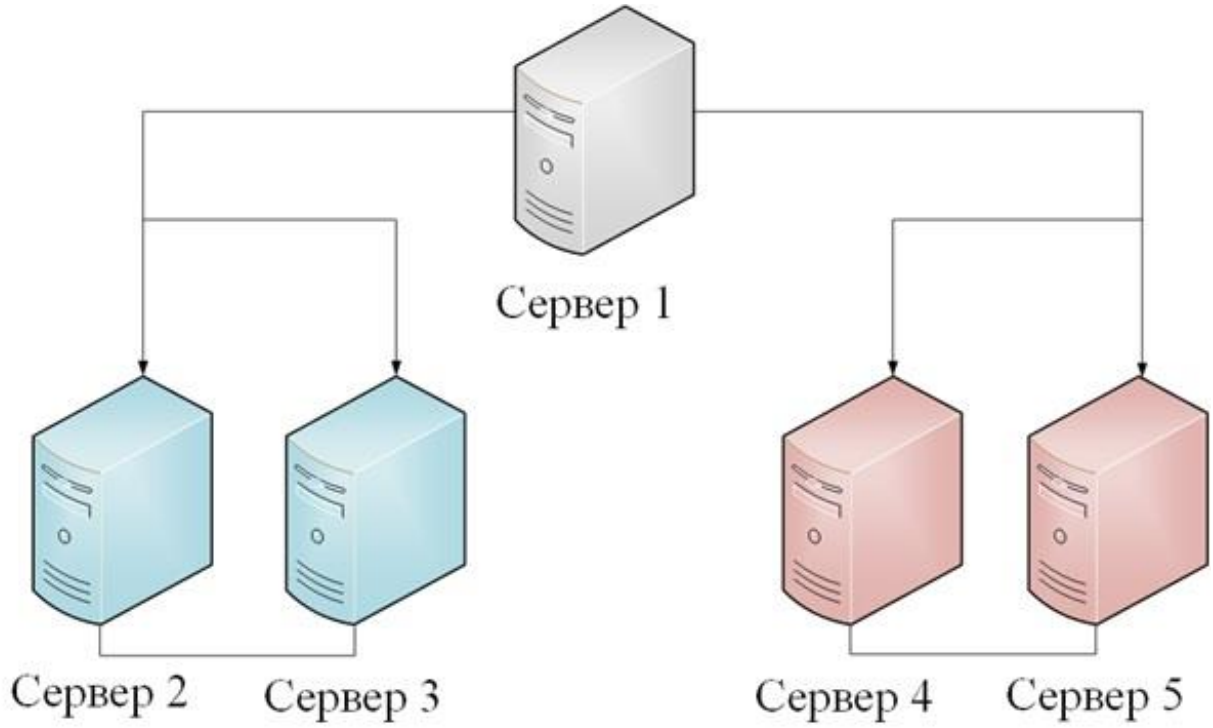

Рисунок 1. Топология кластера для СУБД ClickHouse и Apache Zookeeper

Сервер 1 обеспечивает обработку данных, занимается распараллеливанием задач между репликами. Он собирает данные присланные от подчиненных серверов и формирует ответ на запрос. На этом сервере данные не хранятся.

Серверы 2,3 хранят первую половину данных. Оба сервера хранят одинаковые данные и постоянно синхронизируются. Эти серверы обеспечивают 1-ый шард.

Серверы 3,4 хранят вторую половину данных. Оба сервера хранят одинаковые данные и постоянно синхронизируются. Эти серверы обеспечивают 2-ой шард.

На каждом сервере в файле etc/hosts для удобства прописываются соответствующие IP адреса серверов и задаются логические имена. Пусть это будут например:

Сервер 1 - clickhouse-master

Сервер 2 - clickhouse-replica1-slave1

Сервер 3 - clickhouse-replica2-slave1

Сервер 4 - clickhouse-replica1-slave2

Сервер 5 - clickhouse-replica2-slave2

Также, на каждый из серверов должны быть внесены определенные изменения в конфигурационный файл config.xml с соответствующими подстановками, которые прописываются в файле metrika.xml. На рисунке 2 приведено описание содержимого файла metrika.xml для сервера 1.

На этом сервере будет создана распределяющая таблица, которая не будет хранить никаких данных, а будет заниматься их распределением.

```
<?xml version = “1.0”?>
<yandex>
< clickhouse_remote_servers >
        <my_cluster>
                <shard>
                        <weight>l</weight>
                        < internal_replication >fals e</internal_replication>
                        <replica>
                                <host>clickhouse-replical-slavel</host>
                                < port>8000</port>
                        </replica>
                        <replica>
                                <host>clickhouse-replica2-slavel</host>
                                <port>8000</port>
```

```
                </replica>
            </shard>
            <shard>
                <weight>1</weight>
                < internal_replication >false</internal_replication>
                <replica>
                    <host>clickhouse-replical-slave2</host>
                    <port>8000</port>
                </replica>
                <replica>
                    <host>clickhouse-replica2-slave2</host>
                    <port>8000</port> </replica>
                </replica>
            </shard>
</my_cluster>
</clickhou se_remote_servers >
</yandex>
```

Рисунок 2. Содержимое файла metrika.xml

Из содержимого файла видно, что имеется 2 шарда, на каждом из которых по 2 реплики: серверы 2,3 и серверы 4,5.

Где:

my_cluster – имя кластера.

weight – вес шарда, данные будут распределяться по шардам в количестве, пропорциональном весу шарда. В рассматриваемом примере данные будут распределены равномерно меду двумя шардами.

host – это адрес удалённого сервера. Может быть указан в виде доменого имени, или IPv4 или IPv6 адресов. В данном случае указывается имя сервера, которое заранее было прописано в ect/hosts.

port – TCP-порт для межсерверного взаимодействия. В данном случае номер TCP-порта, прописанного в config.xml равен 8000.

На рисунке 3 приведено содержимое файла metrika.xml для сервера 2.

```
<?xml version = “1.0”?>
<yandex>
<zookeeper-servers>
        <node index = “1”>
        <host> clickhouse-master host </host>
        <port> 2181 </port>
        </node>
</zookeeper-servers>
<macros>
        <layer>1</ layer>
        <shard>1</ shard>
        <replica> replica_slave1</replilica>
</macros>
< clickhouse_remote_servers >
        <my_cluster>
                <shard>
                        <weight>l</weight>
                        < internal_replication >fals e</internal_replication>
                        <replica>
                                <host>clickhouse-replical-slavel</host>
                                < port>8000</port>
                        </replica>
                        <replica>
```

```
                    <host>clickhouse-replica2-slavel</host>
                    <port>8000</port>
                </replica>
            </shard>
            <shard>
                <weight>1</weight>
                < internal_replication >false</internal_replication>
                <replica>
                    <host>clickhouse-replical-slave2</host>
                    <port>8000</port>
                </replica>
                <replica>
                    <host>clickhouse-replica2-slave2</host>
                    <port>8000</port> </replica>
                </replica>
            </shard>
</my_cluster>
</clickhou se_remote_servers >
</yandex>
```

Рисунок 3. Содержимое файла metrika.xml для сервера 2

Заметим то, что на данном сервере будет создана реплицируемая таблица, которая является одной из реплик 1-ого шарда. Поэтому, прописывается адрес ZooKeeper кластера. Это будет IP адрес сервера 1. Также укажем секцию macros, из которой будут доставаться подставляемые значения при создании таблицы с движком такого типа.

Для серверов 3, 4 и 5 файл metrica.xml выглядит аналогичным образом, отличие состоит лишь в изменении номера шарда в секции macros для серверов 4 и 5, поскольку они образуют 2-ой шард.

Далее, когда все необходимые настройки и конфигурации совершены, можно создавать самим таблицы. На рисунке 4 приведен синтаксис создания распределяющей таблицы на сервере 1.

```
CREATE TABLE inbound_call_data (...)
ENGINE = Distributed (my_cluster, default, inbound_call_data, rand ())
```

Рисунок 4. Синтаксис создания таблицы на сервере 1.

Таблица на сервере 1 создается со специальным модулем **Distributed**. Ниже приведены параметры, указываемые при создании таблицы с таким модулем:

my_cluster – имя кластера в конфигурационном файле сервера 1;

default – имя базы данных по умолчанию;

inbound_call_data – наименование создаваемой таблицы;

rand() –генерация ключа шардирования при помощи генератора случайных чисел .

На рисунке 5 приведен синтаксис создания репликационных таблиц на все серверах

```
CREATE TABLE inbound_call_data (...)
ENGINE = ReplicatedMergeTree ('/clickhouse/tables/{layer}-{shard}/inbound_call_data', '{replica}', created_ts_date, (sipHash64(contact (project_id,session_id)), created_ts), 8192)
```

Рисунок 5. Синтаксис создания таблицы на серверах 2, 3, 4 и 5.

Таблицы на серверах 2, 3, 4 и 5 создаются с модулем ReplicatedMergeTree. Ниже указаны параметры, которые прописываются при создании таблицы таким образом:

- **/clickhouse/tables/{layer}-{shard}/inbound_call_data** – путь к таблице в Apache ZooKeeper
- **{replica}** – имя реплики в Apache ZooKeeper

Как видно, эти параметры могут содержать подстановки в фигурных скобках. Подставляемые значения достаются из конфигурационного файла, из секции macros. Описание конфигурационного файла и секции macros приведено выше.

Далее указываются следующие параметры:

- **created_ts_date** – имя столбца типа Date, содержащего дату;
- **(sipHash64(concat(project_id,session_id)),created_ts)** – кортеж, определяющий первичный ключ таблицы;
- **8192** – гранулированность индекса.

Заключение

После выполнения проведенного эксперимента, можно сделать вывод что СУБД Yandex ClickHouse довольна не сложная в ее инсталляции, настройка и конфигурация занимают приемлемое время. Хорошо настроенное логирование СУБД Yandex ClickHouse позволяет быстро отладить систему, что очень удобно при использовании кластера из большого количества серверов. Из минусов ClickHouse можно отнести возможность установки только на сервера Ubuntu 12.04, 14.04, 16.04 версий, а также зависимость от стороннего сервиса Apache Zookeeper, на который ложится ответственность за репликацию и шардирование данных.

Использованные источники

1. Руководство Yandex ClickHouse. Официальный сайт. [Электронный ресурс]. Режим доступа: https://clickhouse.yandex/reference_ru.html (дата обращения: 12.01.2018).
2. Руководство Apache ZooKeeper. Официальный сайт. [Электронный ресурс]. Режим доступа: https://zookeeper.apache.org/doc/r3.5.2-alpha/ (дата обращения: 12.01.2018).
3. Flavio Junqueira, Benjamin Reed. ZooKeeper Distributed Process Coordination. USA, O'Reilly Media, 2014, 246 p.