УДК 336.203

РАБОТА С БОЛЬШИМИ ДАННЫМИ И БАЗОЙ ДАННЫХ В СЕТИ ИНТЕРНЕТА ВЕЩЕЙ

В.А. ВИШНЯКОВ, С.К. ЭЛЬ ХАДЖИ

Белорусский государственный университет информатики и радиоэлектроники, Республика Беларусь

Поступила в редакцию 9 ноября 2020

Аннотация. Представлены десять характеристик сущности больших данных (БД) для работы в сетях интернета вещей (ИВ). Рассмотрены особенности не реляционных баз данных и приведены характеристики трех наиболее распространенных из них: Apache Cassandra, MongoDB, HBase. На примере БД Cassandra даны концепции для хранения больших данных. Рассмотрены ключевые различия в моделировании данных для БД Cassandra по сравнению с реляционной базой данных. Приведен пример построения БД Cassandra для обработки в сети ИВ информации от десяти датчиков качества продукции.

Ключевые слова: большие данные, не реляционные базы данных, сеть ИВ.

Введение

Большие данные – это область приложений, которая рассматривает способы анализа, систематического извлечения информации из наборов данных, которые слишком велики или сложны для обработки традиционными (реляционными) системами управления базами данных.

Большие данные обладают специфическими характеристиками и свойствами, которые могут помочь понять как проблемы, так и преимущества инициатив в области больших данных. В работе [1] авторы представляют свою модель больших данных − 10V, в которой обсуждают пять дополнительных характеристик, добавляя их к пяти, ранее известных, для описания больших данных. Рассмотрим все десять характеристик.

1. Объем. Является наиболее известной характеристикой больших данных.

2. Скорость. Скорость – это быстрота, с которой данные генерируются, производятся, создаются или обновляются.

3. Разнообразие. Когда речь заходит о больших данных, необходимо обрабатывать не только структурированные данные, но и полу структурированные данные.

4. Изменчивость. Первое – это количество несоответствий в данных. Большие данные также изменчивы из-за множества измерений данных, возникающих из множества различных типов и источников данных. Изменчивость может относиться к несогласованной скорости, с которой большие данные загружаются в базу данных.

5. Правдивости. По мере увеличения любого или всех вышеперечисленных свойств достоверность (уверенность или доверие к данным) падает.

6. Действия. Как и достоверность, валидность относится к тому, насколько точны и корректны данные для их предполагаемого использования. Поэтому необходимо принять эффективные методы управления данными, чтобы обеспечить согласованное качество данных, общие определения и метаданные.

7. Уязвимость. Большие данные порождают новые проблемы безопасности. В конце концов, нарушение данных с большими данными – это большое нарушение. Сбор и хранение больших данных часто приводит к утечкам данных.

8. Волатильность. Волатильность относится к принятию решения о том, сколько лет должны быть данные, прежде чем они будут считаться неактуальными, историческими или бесполезными больше, и как долго данные должны храниться.

9. Визуализация. Еще одна особенность больших данных заключается в том, насколько сложно их визуализировать. Современные средства визуализации больших данных сталкиваются с техническими проблемами из-за ограничений технологии in-memory и плохой масштабируемости, функциональности и времени отклика.

10. Значение. Другие характеристики больших данных бессмысленны, если не извлекается из них бизнес-ценность. В больших данных можно найти существенную ценность, оптимизировать процессы и улучшать производительность машин или бизнеса.

Базы данных NoSQL

Базы данных NoSQL (не SQL или не только SQL) стали стандартной платформой данных и основной промышленной технологией для работы с огромным ростом объема данных. Концепция баз данных NoSQL [2] была предложена для эффективного хранения и обеспечения быстрого доступа к большим наборам данных, объем, скорость и изменчивость которых трудно решить с помощью традиционных систем управления реляционными базами данных. Базы данных NoSQL используют различные структуры данных (например, ключ-значение, широкий столбец, документ, график и т.д.) По сравнению с табличными реляционными базами данных эти базы данных не содержат схем, поддерживают легкую репликацию, имеют простой API, согласованы и могут обрабатывать огромные объемы данных.

На рынке существует множество баз данных NoSQL, различные отраслевые тенденции свидетельствуют о том, что Apache Cassandra входит в тройку лучших используемых сегодня вместе с MongoDB и HBase [3].

Apache Cassandra – это распределенная и децентрализованная система хранения данных с открытым исходным кодом для управления очень большими объемами структурированных данных, распределенных по нескольким центрам обработки данных. Она обеспечивает высоко доступное обслуживание без единой точки отказа.

Apache HBase – это не реляционная распределенная база данных с открытым исходным кодом, созданная по образцу BigTable от Google и написанная на Java. Она разработан как часть проекта Apache Hadoop и работает поверх HDFS, предоставляя BigTable-подобные возможности для Hadoop.

MongoDB – это кроссплатформенная документо-ориентированная система баз данных, которая не используем традиционную табличную структуру реляционных баз данных в пользу JSON-подобных документов с динамическими схемами, что упрощает и ускоряет интеграцию данных в определенные типы приложений.

Гипотеза CAP [4] определяет компромисс между доступностью системы, согласованностью и допуском разбиения, утверждая, что только два из трех свойств могут быть сохранены в распределенных реплицированных системах одновременно.

База данных Cassandra

Cassandra предлагает надежную поддержку кластеров, охватывающих несколько центров обработки данных, с асинхронной репликацией без мастера, позволяющей выполнять операции с низкой задержкой (особенно для записи/обновления) [5].

Согласованность в Cassandra может быть настроена таким образом, чтобы обеспечить компромисс между доступностью и задержкой в сравнении с точностью данных. Уровень согласованности чтения определяет, сколько узлов реплики должны отвечать на запрос. Уровень согласованности записи определяет количество реплик, на которых запись должна завершиться успешно, прежде чем клиент получит подтверждение. Cassandra поддерживает два типа операций записи с небольшой разницей между ними: insert и update. Эта БД использует следующие концепции для хранения данных.

1. Кластер – это совокупность узлов или центров обработки данных, расположенных в кольцевой архитектуре. База данных Cassandra распределена по нескольким машинам, которые работают вместе. Самый внешний контейнер называется кластером. Для обработки сбоев каждый узел содержит реплику, и в случае сбоя реплика берет на себя ответственность. Кассандра упорядочивает узлы в кластере в кольцевом формате и назначает им данные.

2. Перебора всех вариантов. Пространство ключей – это самый внешний контейнер для данных в Cassandra.

3. Семейство столбцов – это контейнер для упорядоченной коллекции строк (строка – единица репликации). Каждая строка, в свою очередь, представляет собой упорядоченную коллекцию столбцов. Семейства столбцов в Cassandra подобны таблицам в реляционных базах данных. Каждое семейство столбцов содержит коллекцию строк, которые представлены картой <RowKey, SortedMap<ColumnKey, ColumnValue>>. Ключ дает возможность получить доступ к связанным данным вместе; в отличие от реляционных таблиц схема семейства столбцов не является фиксированной, и Cassandra не заставляет отдельные строки иметь все столбцы. Пользователь может свободно добавлять любой столбец в любое семейство столбцов в любое время.

Семейство столбцов Cassandra имеет следующие атрибуты:

– keys_cached − он представляет собой количество местоположений, которые будут храниться в кэше для каждого SSTable;

– rows_cached − представляет собой количество строк, все содержимое которых будет кэшироваться в памяти;

– preload_row_cache − указывает, хотите ли вы предварительно заполнить кэш строк.

4. Колонки. Столбец – это базовая структура данных Cassandra с тремя значениями, ключ или имя столбца, значение и отметка времени; это единица хранения в Cassandra. Столбцы и количество столбцов в каждой строке могут отличаться в отличие от реляционной базы данных, где данные хорошо структурированы.

5. Суперколонки. Суперстолбец – это специальный столбец, поэтому он также является парой ключ-значение. Но суперколонка хранит карту подколонок, как правило, семейства столбцов хранятся на диске в отдельных файлах.

6. Коллекции. В отличие от концепций внешних ключей и соединений, используемых реляционными базами данных, отношения в Cassandra представлены с помощью коллекций. Cassandra избегает объединения между двумя таблицами, сохраняя адреса электронной почты пользователя в столбце коллекции в таблице user. Каждая коллекция определяет тип хранящихся данных и может быть одной из следующих трех: набор, список, карта.

БД использует язык Cassandra Query Language (CQL) как простой интерфейс для доступа, который является альтернативой традиционному языку структурированных запросов (SQL), используемому СУБД. CQL добавляет слой абстракции, который скрывает детали реализации этой структуры и предоставляет собственные синтаксисы для коллекций и других распространенных кодировок. Языковые драйверы доступны для Java (JDBC), Python (DBAPI2), Node.JS (Helenus), C++ и т.д.

Основные правила моделирования данных в БД Cassandra

Ключевые различия в моделировании данных для Cassandra по сравнению с реляционной базой данных включают следующее [6].

1. Не присоединяется. Нельзя выполнять соединения в Cassandra. Если при разработке модели данных, нужно что-то вроде соединения, придется либо выполнить работу на стороне клиента, либо создать денормализованную вторую таблицу, представляющую результаты соединения.

2. Отсутствие ссылочной целостности. Хотя Cassandra поддерживает такие функции, как облегченные транзакции и пакеты, в ней нет понятия ссылочной целостности между таблицами. В реляционной базе данных можно указать внешние ключи в таблице, чтобы ссылаться на первичный ключ записи в другой таблице, такие операции, как каскадное удаление, недоступны.

3. Денормализация. При проектировании реляционных баз данных важно понятие нормализации. Это не является преимуществом при работе с Cassandra, она работает лучше всего, когда модель данных денормализована.

4. Запрос. Реляционное моделирование, означает, что работа начинается с концептуальной области, а затем представляются сущности в этой области в таблицах. Затем назначаются первичные ключи и внешние ключи для отношений в модели. Когда есть отношение «многие ко многим», создаются таблицы соединений, которые представляют только эти ключи. Таблицы

соединений не существуют в реальном мире и являются необходимым побочным эффектом работы реляционных моделей. В Cassandra начинается работа не с модели данных, а с модели запросов. С помощью Cassandra моделируются запросы и данные должны быть организованными вокруг них.

5. Проектирование для оптимального хранения. Поскольку таблицы Cassandra хранятся в отдельных файлах на диске, важно, чтобы связанные столбцы определялись вместе в одной таблице.

6. Сортировка дизайнерское решение. В Cassandra сортировка трактуется иначе; это дизайнерское решение. Порядок сортировки, доступный для запросов, является фиксированным и полностью определяется выбором столбцов кластеризации, предоставленных в команде CREATE TABLE. Оператор CQL SELECT поддерживает порядок по семантике, но только в порядке, указанном столбцами кластеризации.

Разработка базы данных Cassandra для сети Интернета вещей

Система IoT включает в себя 10 датчиков качества продукции (температура). Каждый из датчиков посылает данные 1 раз в секунду. Таким образом, основные требования можно сформулировать следующим образом:

– 1 система должна продолжать запись, если один узел перестает работать;

– 2 система должна записывать 10 новых записей в секунду, несмотря на возможные сбои узла/сети;

– 3 система должна сообщать обо всех измерениях, собранных любым датчиком за указанный день в течение нескольких миллисекунд;

– 4 система должна как можно быстрее сообщать обо всех измерениях, собранных любым датчиком в течение указанного периода времени.

Проектирование базы данных. Cassandra выполняет запись гораздо быстрее, чем другие базы данных SQL и NoSQL, поэтому она может быть идеальным выбором для работы с десятками запросов на запись в секунду. Кроме того, для удовлетворения требований надежности разработчики могут выбрать мультиреплицированный режим.

Архитектура кластера Cassandra (например, коэффициент репликации = 3), в которой один из узлов развернут в облаках. Для выполнения третьего требования можно принять решение о хранении всех значений, собранных в течение одного дня, в одном необработанном виде. Для этого нам нужно создать составной первичный ключ, который состоит из времени события используемого в качестве ключа кластеризации, а также составного ключа раздела, включая текущую дату и идентификатор датчика.

```
CREATE TABLE temperature_events_by_day (
day text,
sensor_id uuid,
event_time timestamp,
temperature double,
PRIMARY KEY ((day, sensor_id), event_time)
)
WITH CLUSTERING ORDER BY event_time DESC;
```

В этом фрагменте day – это текст в формате: «ГГГГ-ММ-ДД»; (day, sensor_id) – представляет собой комбинированный ключ (формируются по дате показания всех датчиков); event_time – это ключ кластеризации (в кластере – значения датчиков в конкретный момент времени в секундах). Благодаря обратной сортировке внутри строки (с порядком кластеризации по event_time DESC) можно получать самые важные (последние) данные всех датчиков.

Ключ раздела используется для идентификации раздела или узла в кластере, в котором хранится эта строка. Когда данные считываются или записываются из кластера, для вычисления хэш-значения ключа раздела используется функция Partitioner. Это хэш-значение используется для определения узла/раздела, содержащего эту строку.

Ключ кластеризации предназначен для хранения данных строк в отсортированном порядке. Сортировка данных осуществляется по столбцам, которые входят в ключ

кластеризации. Такое расположение позволяет эффективно извлекать данные с помощью ключа кластеризации.

Поиск ключа раздела, который является day/sensor_id, является очень быстрой операцией в Cassandra. Таким образом, предлагаемая модель данных удовлетворяет третьему требованию.

Для реализации четвертого требования потребуется поиск определенного дня (дней) на первом этапе и сравнения временных меток внутри каждого дня на втором этапе. Однако это может занять некоторое время, если база данных включает собранные данные в течение нескольких дней. Учитывая тот факт, что в системе всего 10 датчиков можно рассмотреть возможность создания второго ключевого пространства, где каждая строка соответствует определенному датчику, игнорируя дни.

```
CREATE TABLE temperature_events_by_day (
sensor_id uuid,
event_time timestamp, temperature double,
PRIMARY KEY (sensor_id, event_time)
)
WITH CLUSTERING ORDER BY event_time DESC;
```

В этом фрагменте sensor_id – это ключ раздела (формируются показание конкретного датчика); event_time – это ключ кластеризации (в строке – значение датчика в конкретный момент времени).

Благодаря обратной сортировке внутри строки (с порядком кластеризации по event_time DESC) можно получать самые важные (последние) данные конкретного датчика.

Заключение

1. Представлены десять характеристик сущности больших данных (БД) для работы в сетях интернета вещей (ИВ). Рассмотрены особенности не реляционных баз данных и приведены характеристики трех наиболее распространенных из них: Apache Cassandra, MongoDB, HBase. На примере БД Cassandra даны концепции для хранения данных, такие как кластер, набор ключей, семейство столбцов, колонки, суперколонки, коллекции.

2. Рассмотрены ключевые различия в моделировании данных для Cassandra по сравнению с реляционной базой данных, включающие соединения, ссылки, денормализацию, запрос, хранение и сортировку.

3. Рассмотрен пример построения БД Cassandra для обработки в сети ИВ информации от датчиков качества продукции. Приведены элементы работы с БД, используя язык Cassandra Query Language.

WORKING WITH BIG DATA AND DATABASE IN THE INTERNET OF THINGS

U.A. VISHNYAKOU, S.K. EL-HAJJ

Abstract. Ten characteristics of the big data entity (DB) for working in the Internet of things (IoT) networks are presented. The features of non-relational data bases are considered and the characteristics of the three most widely distributed such databases are given: Apache Cassandra, MongoDB, and HBase. Using the example of Cassandra DB, concepts for storing big data are given. Key differences in data modeling for the Cassandra DB compared to a relational database are considered. An example of building the Cassandra data base for processing information from ten product quality sensors in the IoT network is given.

Keywords: big data, non-relational data bases, IoT network.

Список литературы

1. Firican G. The 10 Vs of Big Data. 8 February 2017. [Electronic resource]. URL: https://tdwi.org/articles/2017/02/08/10-vs-of-big-data.aspx.
2. Evans E. NoSQL 2009. 12 May 2009. [Electronic resource]. URL: http://blog.sym-link.com/2009/05/12/nosql_2009.html.
3. Github. Benchmarking Cassandra and other NoSQL databases with YCSB [Electronic resource]. URL: https://github.com/cloudius- systems/osv/wiki/Benchmarking-Cassandra-and-other-NoSQL-databases-with-YCSB.
4. Brewer E. Towards Robust Distributed Systems in 19-th Annual ACM Symposium on Principles of Distributed Computing, Portland, USA, 2000.
5. DataStax. Apache Cassandra 2.1 for DSE. About data consistency. 14 February 2018. [Electronic resource]. URL: https://docs.datastax.com/en/cassandra/2.1/cassandra/dml/dmlAboutDataConsi stency.html.
6. Carpenter J., Hewitt J. Cassandra: The Definitive Guide. O'Reilly Media, 2016.