Műszaki Tudományos Közlemények vol. 9. (2018) 83–86.
https://doi.org/10.33894/mtk-2018.09.16
Hungarian: https://doi.org/10.33895/mtk-2018.09.16,
http://eda.eme.ro/handle/10598/30297

OVERVIEW OF MODERN NOSQL DATABASE MANAGEMENT SYSTEMS. CASE STUDY: APACHE CASSANDRA

Katalin FERENCZ

Sapientia Hungarian University of Transylvania, Faculty of Technical and Human Sciences, Department of Electrical Engineering, Târgu-Mureş, Romania, ferenczkatalin@yahoo.com

Abstract

The wide spread of IoT devices makes possible the collection of enormous amounts of sensor data. Traditional SQL (structured query language) database management systems are not the most appropriate for storing this type of data. For this task, distributed database management systems are the most adequate. Apache Cassandra is an open source, distributed database server software that stores large amounts of data on low-coast servers, providing high availability. The Cassandra uses the gossip protocol to exchange information between the distributed servers. The query language used is the CQL (Cassandra Query Language).
In this paper we present an alternative solution to traditional SQL-based database management systems - the so called NoSQL type database management systems, summarize the main types of these systems and provide a detailed description of the Apache Cassandra open source distributed database server installation, configuration and operation.

Keywords*: database, NoSQL, distributed, Cassandra, CQL.*

1. Introduction

The most important feature of today's rapidly evolving world is keeping track of the surrounding environment, collecting, storing and processing logged information. We use different intelligent sensors and tools connected to the Internet to map our environment. They are constantly sending data that is stored somewhere in the "cloud" from the users' point of view. However, for professionals developing such applications, the effective design and practical implementation of "cloud storage" is a major challenge.

During our research, the main goal is to set up a distributed database system with multiple computers and logging sensor data for various IoT devices. In practice, we also want to demonstrate that an IT system can be developed that is capable of the storageing and appropriate processing of large amounts of data.

2. Database management systems

Before the fast development phase of IoT was launched, the most common database management systems were SQL-based systems. However, this development also brings the development of database management systems. So in the early 2000s began the development of the NoSQL databases and NoSQL (Not Only SQL) query language that differs from SQL-based databases in storing and querying data. The NoSQL can be referred to as a shared database and solves the problem of SQL databases; that data can only be stored on a single computer, and gives the opportunity to store it in the cloud. It is a very important feature that they have a dynamic schema for unstructured data and can be scaled horizontally, which means that we can connect new machines and servers to handle better the increased traffic.

In the case of NoSQL databases there are four different methods that can be used to store the data:
– key/value store, which stores the keys and their assigned values;
– document store, where semi-structured data can be stored;
– column store, where the data columns are stored one after the other inside a table;
– graph store, in which the data can be well-modelled as a graph and the data are linked with an indefinite number of connections.

With the general spread of the Internet, and with the emergence of sensors, we live in a Big Data period, when large numbers of users can create and access huge amounts of information. This is why Big Data applications present three challenges for which NoSQL technology tries to provide the solution. This problem is commonly referred to as 3V that stands for: volume – velocity – variety.

NoSQL type systems give up immediate consistency (consistency, contradiction-exemption) because their main concept is to have more computers assigned to a database. If a server fails, it switches to another server, ensuring maximum availability.

The CAP theory tries to describe the NoSQL properties: consistency – availability – partition tolerance. [1]

The most common NoSQL database management systems used today are summarized in **Table 1.**

Table 1. *The most common NoSQL database management systems* [2]

Name	Data model	Use
Cassandra	Hybrid of key-value and column-oriened models; Cassandra query language	CERN, eBay, Netflix, GitHub
Redis	Het of (key, value), complex types	Twitter, GitHub, Flickr, Stack Overflow
Voldemort	Complex key-values com-pound objects	LinkedIn
DynamoDB	Document and key-value models	Amazon, BMW
Allegro Graph	RDF-Resource Description Framework, graph database	Stanford, IBM, Ford, Siemens, NASA
Memcached	No replication and persistence	Wikipedia

Among the database management systems included in the table, the Apache Cassandra database management system is used in this research, because it is widespread, open source, well-documented and has many APIs already developed.

3. Apache Cassandra

Apache Cassandra is an open-source distributed database server software, developed in the C/C++ programming language, whose purpose is to store large amounts of data on low-cost servers and provide high availability. The development of Cassandra, known as the developer of Amazon DynamoDB, started with Avinash Lakshman and Prashant Malik on Facebook, so it is not surprising that it follows the Amazon DynamoDB architecture and vision. The project is a member of the Apache Incubator from March 2019. [3]

Cassandra supports the operation of clusters in multiple data centers. The data model used in its structure is a hybrid of keyvalue and column data model. There are two key points for a Cassandra system: the data partition and the data model.

Due to data security, data is stored on multiple machines, so synchronizing data between the machines is very important.

The main features of Apache Cassandra are:
– it is a distributed system: there are many computers in the network and these are connected;
– it is decentralized: this is a very important feature of Cassandra, because it distinguishes it from other database management systems. Accordingly, it is not masters-slave based, there is no Single Point Failure, so if a port is damaged, the entire system will not collapse and the data will be available. This is due to the fact that there is a ring topology, there are data centers, and every node within a data center has the same value;
– it is fault-tolerant: the data is stored on at least 3 computers and there is no data loss;
– it has high availability;
– it is flexible scalable: if you increase the burden on a system and expand it, it can be done without notification by adding a new node;
– the linear scalability: if the number of nodes is doubled, the database server's permeability is doubled;
– tuneable consistency: at the expense of availability we can enhance consistency.

A Cassandra cluster can be made from multiple physical computers and from multiple virtual machines that are networked. Since there are

other computers in the network, members of the cluster will know which node belongs to the cluster based on the information contained in the configuration files. The cluster members use the Gossip protocol to communicate with each other. It also uses a different protocol to form the cluster configuration: this is the Snitch that allows specifying which node belongs to the data center and forms the cluster.

Once the cluster has been set up using Gossip and Snitch protocol settings, in order for the system to be fault-tolerant and have high availability, it is necessary to define the location of the data so it is necessary to determine which node or nodes to store data in. For high availability, it is necessary that the data be stored on multiple nodes.

Fault-tolerance and high availability mean that we replicate the data, usually three replicas at minimum. This means that we have calculated a given token for the given partition key, which determines which node will hold that partition, but in the cluster ring the next (clockwise) two node (if the replication factor is 3) will also store this partition.

The **Figure 1.** shows that the client writes data to the data center and that the B node is responsible for the data, so the 2 nodes next to it will also store this data. Note that the client application communicates with the F node, so that it is now the coordinator, but it can communicate with any other node, in that case the node would become a coordinator as the nodes are equal.

Another important feature of Cassandra is that we can control the consistency level to determine how many replicas should succeed in writing or reading. By default, you can have three values:
– ONE: it is sufficient if one node feedback is sent;
– ALL: each node must be answered;
– QUORUM: the majority must answer.

Immediate consistency is achieved if the sum of writing and reading consistency levels is greater than the replication factor.

Cassandra Query Language (CQL) is the primary language that allows users to communicate with the Apache Cassandra database. The easiest way to interact with Cassandra is by using the CQL shell, cqlsh. Using this, you can create keyspaces and tables, insert new information, query tables, and many other actions.

CQL offers a variety of built-in data types, including collection types. In addition, it allows the user to create their own unique data types.

4. Summary

As a direct result of our work, we have created a knowledge base that allows easy learning of the CQL query language, understand the operation of the Cassandra NoSQL database management system, and to create a few node-based Cassandra clusters.

Using this documentation developed for educational purposes it can be easy to customize data-

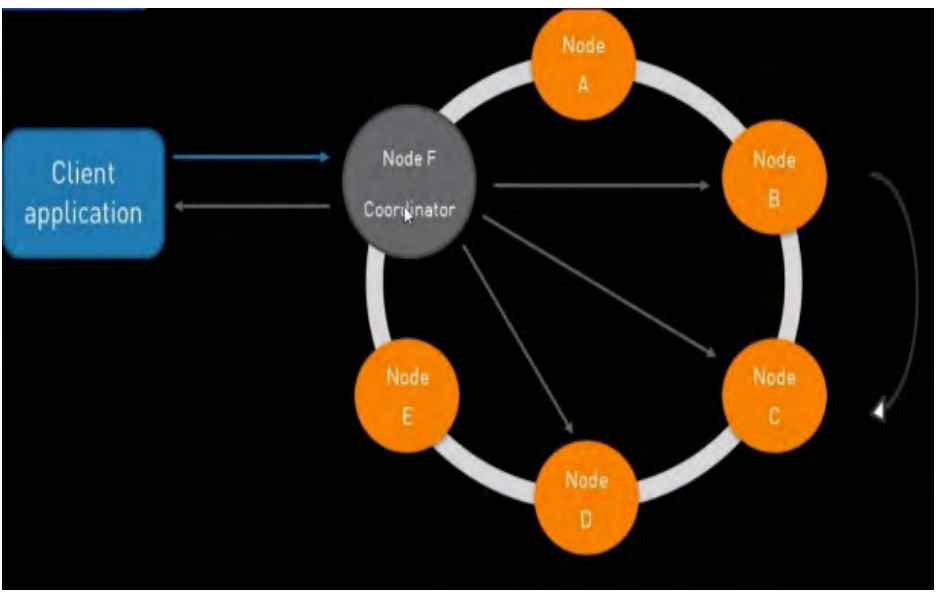

Figure 1. *Cassandra replication*

base operations such as: creation, insertion and querying, modifying, deleting data.

Future work includes the design and development of an electronic circuit for IoT sensor data acquisition and also the storage, visualization and processing of these data.

References

[1] https://en.wikipedia.org/wiki/NoSQL
[2] N.Q.Mehmood, R.Culmone, L. Mostarda: *Modeling temporal aspects of sensor data for MongoDB NoSQL database*, J Big Data(2017) 4:8
[3] http://cassandra.apache.org