Cassandra

Sabyasachi Roy Choudhury[1]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: sabyasachi087@gmail.com

project-000, April 30, 2017

Apache Cassandra is a 'NoSql' database meant to handle a large volume of data through use of commodity hardwares. In this paper we examine Cassandra by understanding the architecture and internal data flows.

Keywords: Cloud, I524

https://github.com/cloudmesh/sp17-i524/raw/master/paper2/S17-IO-3015/report.pdf

1. INTRODUCTION

Apache Cassandra is an open source column-oriented database that bases its distribution design on Amazon's Dynamo and its data model on Google's Bigtable [**?**]cassandra-book. It was developed by Facebook to handle large volume of writes and fault tolerance. The choice of database is solely on the basis of requirements. Cassandra is meant for scalability. If the need is to support thousands of write operations with millions of records, Cassandra or any other column oriented database is much more suitable. But if your need is to support transactions and considerably lower read/write access, RDBMS (Relational Database Management System) is best suited.

1.1. Column Oriented Database

Column Oriented Database 'cite:'www-column-db uses columns to store data tables rather than using rows as in traditional RDBMS. The main difference between column and row approach is schema definition. Column oriented databases have flexibility in column, in which it is not necessary for all the rows have same columns structure, but in RDBMS they are fixed and same for all the rows.

2. TERMINOLOGIES

Partitioning [1]Cassandra is a distributed system where data is distributed across multiple nodes. Each node is responsible for a part of the data.

Replication [1]In a distributed architecture, if one node is down , one of the data source is also sacrificed along with it. To avoid this, data in each node is copied to multiple nodes ensuring fault truculence and resulting in no single point of failure.

Gossip Protocol [2]Since Cassandra is a distributed system, it is important for individual nodes to know the existence and state of each other. To do so , Cassandra uses Gossip protocol.

Memtable [1]It is an In-Memory-Table or a write back cache in which data has not yet flushed into disk.

Column Family [3]It is a NoSql object to store key-value pair. Each column family has one key mapped with set of columns. Each column contains column name , its value and timestamp.

Bloom Filters [1]This algorithm helps to determine of a key is not present in a specific location. This helps in reducing I/O operations.

3. ARCHITECTURE

3.1. Cassandra Cluster/Ring

We have already covered that Cassandra is a distributed system. Each node of the system is assigned with an id or name to uniquely identify it. The set of nodes which helps Cassandra to start up, are know as seeds. Cassandra uses this seed to retrieve informations about other existing nodes. It uses gossip protocol for intra node communication and failure detection. A node exchanges state informations only to other three nodes. This state information contains data about itself and about other known three members reducing IO operations.

3.2. Data Distribution and Replication

Each node of Cassandra, is responsible for a specific range or set of data. During the start-up, every node is assigned with range of token ensuring evenly distribution of data. In figure 1, 0-255 range of data is distributed in four nodes. Hashing technique reviewed earlier is use to create the token of the row key. The row key falling under any of the above shown range will be assigned to its corresponding node. Say for example if

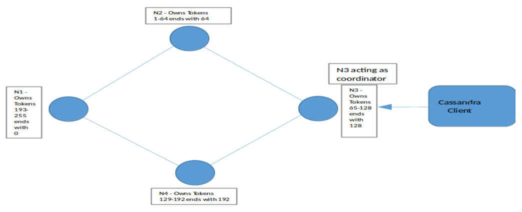
N2 - Owns Tokens 1-64 ends with 64
N3 acting as coordinator
N1 - Owns Tokens 193-255 ends with 0
N3 - Owns Tokens 65-128 ends with 128
Cassandra Client
N4 - Owns Tokens 129-192 ends with 192

Fig. 1. Cassandra Cluster Ring

the hash value of the row key comes to 38 , it will go to the node N2. In a distributed system, once can't rely on a single node for storing a set of data, as if the node is down that particular set won't be available for read, write and update. To achieve a better reliability and fault tolerance , Cassandra replicates data in multiple nodes. It has two basic replication strategy :

1. Simple - In this data is copied on to the next node in a clockwise manner
2. Network-Topology - In this Cassandra is aware of the node's

3.3. Read and Write Paths

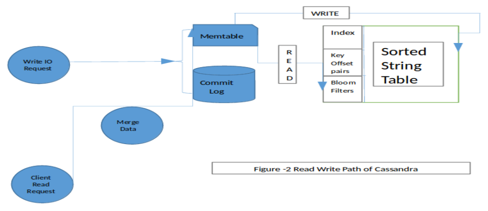

Fig. 2. Cassandra Read Write Data Flow

In figure 1, the client is connected with node 3. Since Cassandra is master less, N3 will be acting as coordinator and will serve for all client requests. It is the responsibility of N3 to communicate with its fellow nodes and fetch the desired results. We have already discussed that not all nodes communicates will all others for data retrieval instead it communicates only with handful of nodes for reducing IO operations. The number of nodes to communicate can be configured using QUORUM or knows as consistency level. Read and write flows has been described in figure 2. Each node processes write requests separately. It writes the data to Commit log first and then to Memtable. In case the node crashes, data can be restored from the Commit log. The data from Memtable will only be flushed to the disk or SSTable if

1. It reaches its maximum allocated size in memory
2. The number of minutes a memtable can stay in memory elapses.
3. Manually flushed by a user

Read operation is similar to write operation. Every read operation must be with row key. As discussed earlier, row-key is used to determine the right node and the request is then passed to that particular node. Read is then catered by the bloom-filter and then proceeds to the desired result set.

4. CHOOSING THE RIGHT DATABASE

4.1. Cap Theorem

Cap Theorem [4] states that, it is impossible for a distributed computer system to simultaneously provide more than two out of three of the following guarantees -

1. Consistency - Every read receives the most recent write or an error.
2. Availability - Every request receives a (non-error) response but doesn't guarantee if the data is recent or not.
3. Partition tolerance - System continues to operate even after loosing (dropped or delayed) an arbitrary number of messages by the nodes.

4.2. Cassandra and Cap Theorem

All Big Data databases are tolerant so considering Cap Theorem one have to choose between Consistency and Availability as per the need. Cassandra is highly available trading off with consistency. As we have seen so far, Cassandra master less architecture allows client to connect any of the node for read write. This increases availability as in case a node is down, the client can quickly jump to nearest available. The disadvantage is while reading data, the data might not be the recent. If the node with recent data us down, the data won't be replicated and reads will lead to the older state. Cassandra is used in write once and read many scenario. For example historical data analysis where the updates are very few.

4.3. Brief Comparison

Apart from Cassandra, Hbase and MongoDB are fairly popular choice of database for big data solutions. Hbase is built on HDFS (Hadoop Distributed File System) and column oriented database similar to Cassandra. Major advantage of Hbase is its querying functionality which is an edge over Cassandra. Hbase can also be installed on already clustered Hadoop environment. Check here for details. Accumulo is another option available which is based upon HDFS as well. The advantage of Accumulo is its cell level security where as all other column oriented databases have column level security. Having security at cell level allows user to see different value as appropriate based on the row. Check here for details. MongoDB stands separate from all of the aforesaid mentioned as it is a document oriented database. Since data fields are to be stored vary between different elements, column oriented storage leads to lot of empty column values. MongoDB provides a way to store only the necessary fields. Check here for more details.

5. CONCLUSION

Cassandra is a column oriented database with focus on high availability. The architecture allows it for fast write operations. Cassandra can handle bulk writes and make it an ideal candidate for storing logs as logs are generally high in volume and speed. Cassandra is poor in table joins and hence not suited for data mining.

ACKNOWLEDGMENTS

I would like to thanks Akhil Mehra for his vibrant description of Cassandra architecture.

REFERENCES

[1] Akhil Mehera / Dzone, "Introduction to cassandra," Web Page, Apr. 2015, accessed: 2015-04-06. [Online]. Available: https://dzone.com/articles/introduction-apache-cassandras

[2] Wikipedia, "Gossip protocol," Web Page, Jan. 2017, accessed: 2017-01-11. [Online]. Available: https://en.wikipedia.org/wiki/Gossip_protocol

[3] Wikipedia, "Column family," Web Page, Jan. 2017, accessed: 2017-01-11. [Online]. Available: https://en.wikipedia.org/wiki/Column_family

[4] Wikipedia, "Cap theorem," Web Page, Jan. 2017, accessed: 2017-04-17. [Online]. Available: https://en.wikipedia.org/wiki/CAP_theorem