Cassandra – A Progressive Database Management System

Durgesh Samariya
M. Tech. Student (Software Technology)
Department of Computer Engineering
VIT University

Abstract

Traditional Relational Database Management systems were the main data stores for most of the business applications for over 20 years. Then in order to handle new data access patters new databases like Oracle, MySQL were introduced that had RDBMS roots. Currently, more change is required since applications must now scale up that were unimaginable just a few years ago. Not only scaling; companies require features like their applications are always available and lightning fast, and this is where RDBMS databases fail. This is where Cassandra is introduced. As for availability, Cassandra delivers a world where an application can lose an entire datacenter and still perform as if nothing happened. In this paper we propose a brief overview of Cassandra for people wondering whether Cassandra is right for them and also uniquely addresses the next phase of growth in the modern database marketplace.

Keywords: Cassandra, RDBMS, Distributed Database

I. INTRODUCTION

Big Data is a phrase which is used to describe a massive collection of structured and unstructured data which can no longer be stored and processed by traditional databases and processing applications. Big Data can be characterized by 3V: Volume (massive amount of data produced), Velocity (speed with which data is generated) and Variety (types of data like structured and unstructured data). The reason behind massive volume of data is that numerous information-sensing devices such as mobiles, cameras, RFID, Software logs, etc, gather it.

The problem that we are facing with big data is that standard tool and procedures that till date we used are no longer capable to process and analyze such massive amount of data. We need altogether a different algorithms and technology to analyze and process Big Data. In order to analyze Big Data first we need to understand what we are looking for, then slicing and dicing of data, simple visualizations and basic monitoring is needed. For advanced analytics we need to have more complex analysis such as predictive modelling and other pattern matching techniques.

Oracle, MySQL, which had RDBMS roots were introduced which were the traditional Relational databases that were acting as the primary data, stores for various business applications. But recently new change is seen in the database management applications, which earlier we termed, as Big Data. Big data which has the characteristics that it is too big in volume, moves very fast i.e. it has velocity no longer fits the structure of RDBMS Architecture. So we need to find the alternative database architecture.

Apache Cassandra, a top level Apache project born at Facebook and built on Amazon's Dynamo and Google's BigTable, is a distributed database for managing large amounts of structured data across many commodity servers, while providing highly available service and no single point of failure. Cassandra offers capabilities that relational databases and other NoSQL databases simply cannot match such as: continuous availability, linear scale performance, operational simplicity and easy data distribution across multiple data centers and cloud availability zones.

Cassandra's architecture is responsible for its ability to scale, perform, and offer continuous uptime. Rather than using a legacy master-slave or a manual and difficult-to-maintain sharded architecture, Cassandra has a masterless "ring" design that is elegant, easy to setup, and easy to maintain.[2]

In Cassandra, all nodes play an identical role; there is no concept of a master node, with all nodes communicating with each other equally[3]. Cassandra's built-for-scale architecture means that it is capable of handling large amounts of data and thousands of concurrent users or operations per second—even across multiple data centers—as easily as it can manage much smaller amounts of data and user traffic. Cassandra's architecture also means that, unlike other master-slave or sharded systems, it has no single point of failure and therefore is capable of offering true continuous availability and uptime — simply add new nodes to an existing cluster without having to take it down.

Many companies have successfully deployed and benefited from Apache Cassandra including some large companies such as: Apple, Comsat, Instagram, Spotify, eBay, Rackspace, Netflix and many more. The larger production environments have PB's of data in clusters of over 75,000 nodes. Cassandra is available under the Apache 2.0 license.[2]

The Apache Cassandra database is the right choice when you need scalability and high availability without compromising performance [1]. Linear scalability and proven fault-tolerance on commodity hardware or cloud infrastructure make it the perfect

platform for mission-critical data. Cassandra's support for replicating across multiple datacenters is best-in-class, providing lower latency for your users and the peace of mind of knowing that you can survive regional outages [1].

Cassandra's data model offers the convenience of column indexes with the performance of log-structured updates, strong support for de-normalization and materialized views, and powerful built-in caching [1].

II. PROPOSED WORK

According to the Gartner group Oracle occupies more than 48% of the market which means that it leads the traditional database technology. Oracle is a solid RDBMS which means it works perfectly for ERP and accounting applications. The main reason why Oracle is replaced by Cassandra is that currently apps are having high velocity data, the data is multi-type which means that either the data can be structured or can be non-structured, write data anywhere,comes from various locations , high data volumes and scales writes and reads.

Today's online system requires a divide-conquer configuration for processing massive and fast data that comes from billions of users. And since it comes from different users hence the data gathered is from different locations. Hence scale-up, master-slave and non-distributed architecture of RDBMS is not designed for such cases.

A. Data Model of Cassandra

Cassandra is designed based upon the CAP Theorem, which is also known as Brewer's theorem.CAP theorem suggest that in a distributed system it is not possible to accommodate all the three properties 1) Availability 2) Consistency 3) Partitiontolerance.

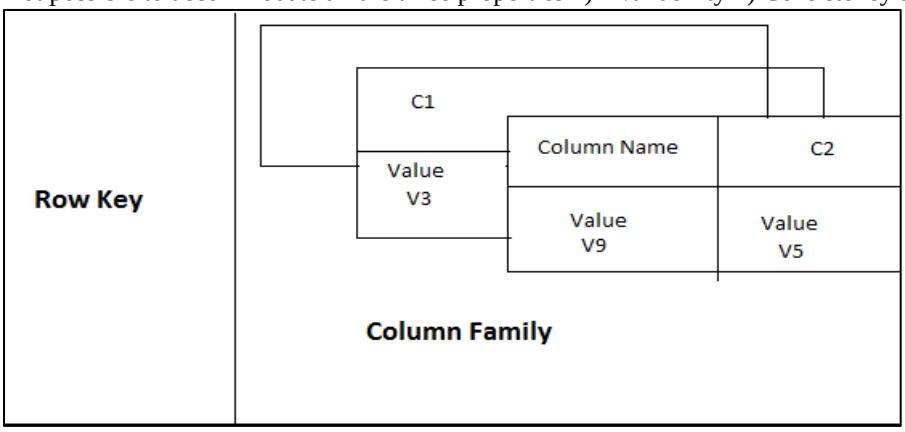

Fig. 1: Data model of Cassandra

Cassandra permits only querying the data based on their respective key. And the columns which does not have key are not indexed. Thus Cassandra Data model consist of the following parameters:

1) Row

Every row group together columns and super-columns.and also row is uniquely identifiable data in the system and is identified by the Row-key.

2) Column-Family

Column-Family group together keyed rows which group columns and super-columns of structured data. Thus column-family is a structured data. All the data stored in column-family must be of same type. Moreover column-names are stored in the form of bytes and do not have size constraint. Efforts should be made that number of different column-family in the keyspace should be as small as possible and no alteration should be made during the operation.

3) Keyspace

Keyspace is the top unit of information.Cassandra takes the general form as (keyspace, column-family, row-key).Thus keyspace can be defined as collection of column-family. Thus according to RDBMS we can consider keyspace as name for schema and column-family name for tables.

4) SSTable

Sorted string table consist of row keys and set of column key/value pairs. Operations are performed in such a way that value is searched for particular associated key and iteration is performed over all the column-names and value pairs within a specific key range. Moreover there is index file that contains index and the start location of the row key. When SStable is opened at that point index file is loaded into the memory so that memory needed for index file can be optimized.

5) MemTable

During any operations like update or delete the data is written in MemTable. Memory table is just temporary location and once it is full the data is transformed to the SStable.

6) Read and Write Operation

While reading the data in Cassandra it merges together the data from MemTable and SStable. Read is performed based on the given row key also the key range is mentioned that specifies the range within which we need to perform reading operation. In

case of multiple updates on the same column Cassandra uses timestamps to resolve the conflicts. For delete operation Cassandra writes the Tombstone to avoid random writes. Tombstone is a special value written to Cassandra instead of removing the data immediately. The tombstone is then sent to those nodes that did not get initial removal request. And then the data is removed.

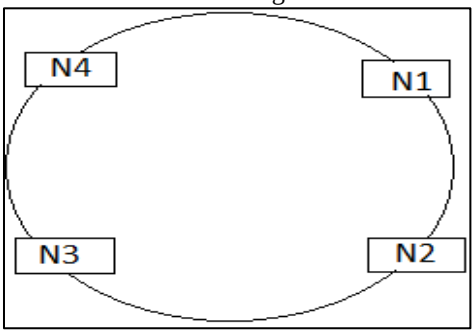

Fig. 2: Ring Architecture of Cassandra

Moreover there is no concept of Master-Slave for the nodes irrespective of what operations it performs. Thus is a benefit of that there is no single point of failure. Cassandra has the feature of scale-out as result new nodes can be easily added. If the data volumes increases then no extra processes are required, Cassandra automatically divides the data across the nodes

Thus from the above architecture it can be seen that progressive database like Cassandra can remove many limitations that were found in RDBMS.

III. CONCLUSION

There's no question that many modern applications have outgrown legacy relational databases. Cassandra has a ring architecture which does not include any master-slave concept. And its architecture is built for scale-up purpose. To handle big data workloads, these systems require a massively scalable NoSQL database. Cassandra is able to handle Velocity, Volume and Variety. While there are a number of NoSQL database providers in the market, only Cassandra is able to offer the linear scale performance and key enterprise-class features that meet the expectations and requirements of big data systems.

REFERENCES

[1] http://cassandra.apache.org/
[2] http://planetcassandra.org/what-is-apache-cassandra/
[3] http://www.datastax.com/products/datastax-enterprise-production-certified-cassandra
[4] Avinash Lakshman ,Prshant Malik "Cassandra –A Decentralized structured storage system", Cornell , 2009.
[5] Mike Burrows "The chubby lock service for loosely-coupled distributed systems" inn OSDI '06.
[6] Dietrich Featherston "Cassandra-Principles and applications".