International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

NoSQL Database: Cassandra is a Better Option to Handle Big Data

Dr. Kishor H. Atkotiya[1], Parag C. Shukla[2]

[1]J.H Bhalodiya Women's College, Rajkot- 360005, India

[2]Atmiya Institute of Technology & Science, Rajkot – 360005, India

Abstract: ***Apache Cassandra is an open source distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers robust support for clusters spanning multiple datacenters,[1] with asynchronous master less replication allowing low latency operations for all clients. Like good carpenters, data engineers know that different tasks require different tools. Picking the right tools -- and knowing how to use them -- can be the most important part of any job. Apache cassandra, a prime level Apache project born at Facebook and designed on Amazon's generator and Google's huge Table, may be a distributed info for managing giant amounts of structured knowledge across several goods servers, whereas providing extremely offered service and no single purpose of failure. cassandra offers capabilities that relative databases and different NoSQL databases merely cannot match such as: continuous handiness, linear scale performance, operational simplicity and simple knowledge distribution across multiple knowledge centers and cloud handiness zones [2]. Cassandra's design is to blame for its ability to scale, perform, and supply continuous time period. instead of employing a bequest master-slave or a manual and difficult-to-maintain shared design, cassandra features a lordless "ring" style that's elegant, simple to setup, and simple to keep up. Apache Cassandra is a massively scalable open source non-relational database that offers continuous availability, linear scale performance, operational simplicity and easy data distribution across multiple data centers and cloud availability zones. Cassandra was originally developed at Facebook, was open sourced in 2008, and became a top-level Apache project in 2010.***

Keywords: NoSQL Database, Cassandra, BigData, BigData Analytics, RDBMS and NoSQL

1. Introduction

As it is understood that cassandra is Associate in Nursing open supply management system that has been notably designed for handling immense quantity of information across several servers with none defaults at any regulated purpose. this is often one in all the explanations why cassandra has been picked up for large knowledge. massive knowledge is needed for managing knowledge and knowledge warehouses. however once the question involves handling immense volumes of information, the information warehouses fails to provide results. there's a demand for a code which may regulate also as grasp immense volumes of information. once a corporation or a corporation expands or amalgamates itself, there ar several cross knowledge and knowledge that needs to be discarded. additionally there ar immense volumes of information that needs to extracted freshly in its raw type. This has been compiled by the cassandra massive knowledge services which may handle these problems terribly simply with none failure or default. massive knowledge solutions has been provided by cassandra massive knowledge services, and one will relate all the options and factors of cassandra and large knowledge.

Cassandra's design is chargeable for its ability to scale, perform, and provide continuous period. instead of employing a bequest master-slave or a manual and difficult-to-maintain shared design, cassandra incorporates a lordless ~~–ri~~ng" style that's elegant, straightforward to setup, and straightforward to take care of.

2. Cassandra Architecture

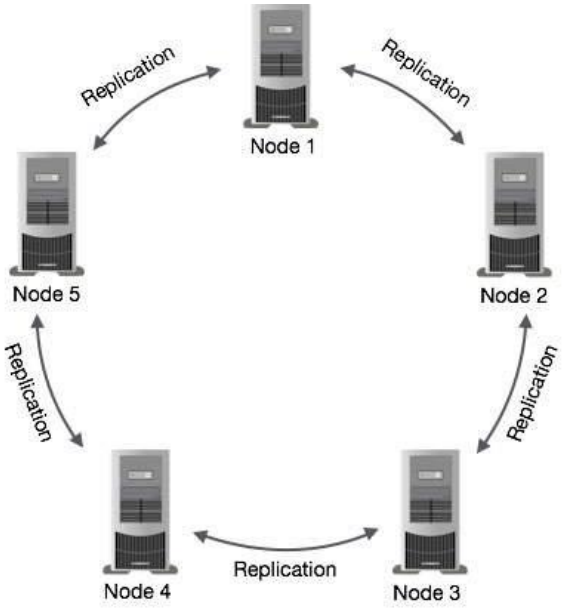
The design goal of cassandra is to handle big data workloads across multiple nodes with none single purpose of failure. prophetess has peer-to-peer distributed system across its nodes, and knowledge is distributed among all the nodes during a cluster.

All the nodes during a cluster play constant role. every node is freelance and at constant time interconnected to different nodes. Each node during a cluster will settle for scan and write requests, in spite of wherever the info is really placed within the cluster. When a node goes down, read/write requests will be served from different nodes within the network.

3. Data Replication in Cassandra

In Cassandra, one or more of the nodes in a cluster act as replicas for a given piece of data. If it is detected that some of the nodes responded with an out-of-date value, Cassandra will return the most recent value to the client. After returning the most recent value, Cassandra performs a read repair in the background to update the stale values.

The following figure shows a schematic view of how Cassandra uses data replication among the nodes in a cluster to ensure no single point of failure



Cassandra uses the Gossip Protocol in the background to allow the nodes to communicate with each other and detect any faulty nodes in the cluster.

4. Cassandra Query Language

Users will access Cassandra through its nodes mistreatment Cassandra command language (CQL). CQL treats the information (Keyspace) as a instrumentality of tables. Programmers use cqlsh: a prompt to figure with CQL or separate application language drivers.

Clients approach any of the nodes for his or her read-write operations. That node (coordinator) plays a proxy between the consumer and also the nodes holding the information.

Write Operations
Every write activity of nodes is captured by the commit logs written in the nodes. Later the data will be captured and stored in the mem-table. Whenever the mem-table is full, data will be written into the SStable data file. All writes are automatically partitioned and replicated throughout the cluster. Cassandra periodically consolidates the SSTables, discarding unnecessary data.

Read Operations
During read operations, Cassandra gets values from the mem-table and checks the bloom filter to find the appropriate SSTable that holds the required data.

5. Features of Cassandra

Always on Architecture — A true masterless architecture (unlike other master/slave RDBMS and NoSQL databases) delivers continuous availability for your applications.

Natively Distributed — The gold standard in multi-data center and cloud replication supplies real write/read anywhere capabilities, allowing you to easily put data where it's needed anywhere in the world.

Fast Linear-Scale Performance — Enables millisecond response times with linear scalability (double your throughput with two nodes, quadruple it with four, and so on) to deliver response time speeds your customers have come to expect.

Flexible Data Model — The Apache Cassandra data model allows for new entities or attributes to be added over time and you're not restricted to a rigid data model that can't evolve with the needs of the business application — such as the addition of a new complicated data structure that may be unique to your environment, or adding a new column to a column family.

Language Drivers — Cassandra supports an incredible array of language drivers to ensure that your application runs optimally on Cassandra – whether Python, C#/.NET, C++, Ruby, Java, Go, and many more.

Operational and Developmental Simplicity — With all nodes in a cluster being the same, there is no complex software tiers to manage so administration duties are greatly simplified. Plus, the Cassandra Query Language (CQL) looks and acts just like SQL, which makes moving to Cassandra from any RDBMS very easy.

Strong Developer Community — There is a rich developer community that surrounds Apache Cassandra that strives to support developers working on the project, as well as those developing applications that leverage the database. Active in the IRC chat room and mailing lists, the Cassandra developer community is one of the most active for an open source project.

6. Cassandra Data Model

On the surface, Cassandra's data model seems to be quite relational. With this in mind, diving deeper into ColumnFamilies, SuperColumns and the likes, will make Cassandra look like an unfinished RDBMS, lacking features like JOINS and most rich-query capabilities.

To understand why databases like Cassandra, HBase and BigTable (I'll call them DSS, Distributed Storage Services, from now on) were designed the way they are, we'll first have to understand what they were built to be used for.

DSS were designed to handle enormous amounts of data, stored in billions of rows on large clusters. Relational databases incorporate a lot of things that make it hard to efficiently distribute them over multiple machines. DSS simply remove some or all of these ties. No operations are allowed, that require scanning extensive parts of the dataset, meaning no JOINS or rich-queries.

There are only two ways to query, by key or by key-range. The reason DSS keep their data model to the bare minimum is the fact, that a single table is far easier to distribute over multiple machines, than several, normalized relations or graphs. Think of the ColumnFamily model as a (distributed Hash-)Map with up to three dimensions. The two-dimensional setup consists of just a ColumnFamily with

some columns in it, –some" meaning a couple of billion if you so wish. So a ColumnFamily is just a map of columns.

Data storage in Cassandra is row-oriented, meaning that all contents of a row are serialized together on disk. Every row of columns has its unique key. Each row can hold up to 2 billion columns. Furthermore, each row must fit onto a single server, because data is partitioned solely by row-key.

One of Cassandra's strengths is high write throughput on commodity hardware, which enables us to scale infrastructure very quickly. Because we handle terabytes of data, a high write rate is critical to us. And because it's hard to predict loads, fast scalability translates into a competitive business advantage.

7. Quick Comparision of RDBMS and NOSQL Cassandra

Relational Database	Cassandra
Handles moderate incoming data velocity	Handles high incoming data velocity
Data arriving from one/few locations	Data arriving from many locations
Manages primarily structured data	Manages all types of data
Supports complex/nested transactions	Supports simple transactions
Single points of failure with failover	No single points of failure; constant uptime
Supports moderate data volumes	Supports very high data volumes
Centralized deployments	Decentralized deployments
Data written in mostly one location	Data written in many locations
Supports read scalability (with consistency sacrifices)	Supports read and write scalability
Deployed in vertical scale up fashion	Deployed in horizontal scale out fashion

8. Conclusion

Cassandra gives us the ability to scale out by simply adding a node to a cluster, and letting the cluster rebalance itself, which saves on operational overhead. Maintaining high write throughput with a minimal number of nodes lets us manage infrastructure costs more effectively. When you're running a business that provides real-time big data analytics, keeping things simple and managing infrastructure costs intelligently are critical objectives.

For IT professionals who are either planning new big data applications under big data workloads, a move to DataStax Enterprise and Cassandra makes both business and technical sense. Switching to a modern, big data platform like DataStax Enterprise will future-proof any application, and provides confidence that the system will scale and perform well both now and into a demanding future.

References

[1] Casares, Joaquin (2012-11-05). "Multi-datacenter Replication in Cassandra". DataStax. Retrieved 2013-07-25. Cassandra's innate datacenter concepts are important as they allow multiple workloads to be run across multiple datacenters.
[2] http://www.planetcassandra.org/what-is-apache-cassandra/
[3] https://en.wikipedia.org/wiki/Apache_Cassandra
[4] "Cassandra is an Apache top level project". *Mail-archive.com. 2010-02-18.* Archived*from the original on 28 March 2010*. Retrieved 2010-03-29.
[5] The Apache Software Foundation Announces Apache Cassandra Release 0.6 : The Apache Software Foundation Blog.
[6] Sylvain Lebresne (10 September 2014). "[VOTE SUCCESS] Release Apache Cassandra 2.1.0". mail-archive.com. Retrieved 11 December 2014.
[7] http://www.planetcassandra.org/blog/usingcassandra-for-real-time-analytics-part-1/
[8] http://www.tutorialspoint.com/cassandra/cassandra_architecture.htm
[9] http://www.informationweek.com/strategic-cio/why-we-picked-cassandra-for-big-data/a/d-id/1318250
[10] http://www.datastax.com/resources/faq
[11] http://blog.markedup.com/2013/02/cassandra-hive-and-hadoop-how-we-picked-our-analytics-stack/