Proceedings of the Second International Conference on Research in Intelligent and Computing in Engineering pp. 33–38
DOI: 10.15439/2017R65
ACSIS, Vol. 10 ISSN 2300-5963

A review of NoSQL Databases and Performance Testing of Cassandra over single and multiple nodes

Sanket Ghule
Department of Information Technology,
Pimpri Chinchwad College of Engineering
Pune, India
ghulesanket7@gmail.com

Ramkrishna Vadali
Department of Information Technology,
Pimpri Chinchwad College of Engineering
Pune, India
ramkrishna.vadali1@gmail.com

***Abstract*—Today in the era of Internet, there is a huge explosion in the size of data. The traditional relational databases are failing to handle the current increase size of the database efficiently, leading to slow data access rate. NoSQL has thoroughly overcome the challenges for handling big data, distributed clusters and the scalability issues. Cassandra is one the leading NoSQL database gaining its importance in various applications. In this paper, Read and write performance tests are carried out which describes that Cassandra improves its performance as the number of cluster increases with increasing access speed.**

***Keywords*—NoSQL; Cassandra; performance testing**

I. Introduction

In the process of development of the data store, database technologies have undergone drastic changes in order to fulfill the user requirements. Therefore traditional relational database has made a remarkable stand in the field of database storage. Coming around the recent years, the increase in the use of Internet Network technology, various modern changes have emerged in applications. Accordingly, some changes were expected in order towards encountering the requirements using the Distributed Data Models. Relational databases are not able compete these requirements. NoSQL (Not Only SQL) is a non-relational database methodology which approaches the increasing changes required to fulfill the requirements.

This paper focuses on NoSQL databases conceptions and technology along with read and writes performance testing on Cassandra, a representative of NoSQL databases. This test is performed for single as well as multi-node scenario.

II. Limitations of SQL

1) Unstructured data:

Relational databases are not capable of managing unstructured data like multimedia files and documents effectively. Although, it can be handled using various methods such as storing the file in file system and just passing a reference to the database or storing the Binary Large Object File (BLOB) in database. But it cannot be done efficiently as it does not have indexes or problems of indexes, unreferenced data, unrecognizable patterns of data, etc.

2) Volume of data:

The complexity in the system increases as the size of database increases (such as in Terabytes and Petabytes), the complexity in the system increases. There is a possibility that this data can be structured, unstructured or both. Therefore Relational database consumes lot of time to process this huge amount of data which hampers the overall efficiency and throughput of the system.

3) Joins:

Relations among the databases are maintained with the help of joins. With the growing size of the database, a complex operation to maintain the joins which leads to slower database access at times. Joins can even make the fastest hardware to slow due to its complexity.

4) Lack of Efficiency and performance:

It is possible to manage large data with the help of relational databases, but cannot be managed in an efficient way. When the database is designed, should be in normalized form; consist of appropriate joins at the right place. It becomes difficult to manage the database while dealing with the large datasets. As the size of the data increases, operations such as data retrieval, searching, aggregations, etc. are increased thus making the database slow that leads to the lack of efficiency.

5) Scalability and Management:

Relational database requires more management as the complexity of databases grows. Database management is required if the database is to be implemented in a distributed approach. Management has to be involved when scalability comes into picture. Administrators have to perform the scaling operation manually which is a complex task to be carried out as all properties of the database must remain intact.

6) Performance:

When the performance of the database falls, it therefore leads to lack of efficiency. Performance degradation can be due to various reasons such as complex batch operations, indexing etc. The performance of the database degrades, as the data size increases.

7) Stability issues:

Relational databases with respect to references and auditing are less reliable. They have weak scalable replication and distribution as they require more management when replicated and/or distributed. It makes it difficult to process long and complicated queries, making it to crash. Therefore in order to run those, it has to be part by part with the stored procedures).

8) User and Query conflicts:

Transactional locks and deadlocks may lead to slow access to database which may frustrate the users.

III. NoSQL (NOT ONLY SQL) DATABASES

NoSQL [1] is a schema less non-relational database. It is quickly growing database over SQL due to factors such as higher scalability, easy replication and distribution of database. NoSQL databases are different from the traditional SQL databases in many aspects such as tabular format, queries, etc. NoSQL does not require pre-designed tabular format as in SQL for storing the data. It is therefore known as unstructured database.

It solves many problems over the traditional relational approach such as distributed, open source, scalable etc. Today there are more than *thirty five* different NoSQL databases available for different purposes, developed for different scenarios.

A. Advantages of NoSQL Databases

NoSQL databases ensures following benefits compared to the traditional ones:

a) Schema Independent: At present, there are few prevalent types of NoSQL databases: Column Store, Document Store, Key Value Store, Graph Store and some other modes. These modes don't require creation of the data fields before usage. Plus, these NoSQL databases do not require fixed table structure which can store custom data formats at any instant of time.

b) Scaling Horizontally: Conventional relational database use scaling up approach to improve performance, while NoSQL databases are utilized to enhance the performance levels by horizontal scaling mode, results in distributing the load equally to each of the host system.

c) Less Management and Low cost: NoSQL databases are open source databases exclusive of expensive licensing charges. This requires less management as most of the operations are carried out through the databases itself. It does not require management as the data goes beyond threshold limit of the data, load is distributed automatically.

e) Easy replication: Easy replication of databases is done in NoSQL thus helping the database to be cooperative in distributed approach.

B. Disadvantages of NoSQL Databases

Evaluated with the conventional relational databases, NoSQL databases even though ensure several benefits, there are various drawbacks too:

a) Complex for beginners: Every NoSQL database has its own query programming, which makes it faster. But users have to be trained in order to work on NoSQL databases as they query differently than traditional SQL, which users are used to be with.

b) Not Reliable: ACID (Atomicity, Consistency, Isolation, and Durability) properties are supported by relational databases; while NoSQL databases won't therefore they do not reach the reliability level that ACID properties provide. If the users require NoSQL databases to employ ACID properties for a dataset, they should perform extra programming.

c) Eventual Consistent: ACID transactions cannot be maintained by NoSQL databases, therefore there are limitations to maintain consistency. Although it supports better performance and scalability still there are complications for specific applications and transactions, for example those comprised with banking. NoSQL databases use this type of consistency which is known as Eventual consistency.

C. Why NoSQL database be choosen

Though NoSQL overcomes traditional SQL databases, but there are several reasons why any user should choose for NoSQL databases:

- NoSQL Databases helps in improving programmer's productivity by using appropriate NoSQL database to match the needs of the application.
- NoSQL databases improve the performance of data access. This is because NoSQL relies on denormalization and is optimized due to the denormalized case. For example, if there is blog containing comments, then comments are stored with blog in NoSQL data stores. This makes the data access faster as the data can be retrieved all together from single location.
- It can handle large data quantity efficiently by not executing any joins or indexing such as in SQL.
- It can helps in reducing the latency period therefore enhancing the overall throughput of the application.

IV. Types of NoSQL databases

D. Key Value Database

The key-value store or key-value database is constructed for storing, retrieving, and supervising hash or dictionary i.e. the data structure. A collection of objects or records are enclosed in these dictionaries, which contains data in many different fields within them. A key is used to store and retrieve these objects or records which uniquely identifies the record, and can be used immediately to locate the data present in the database. [2][3]

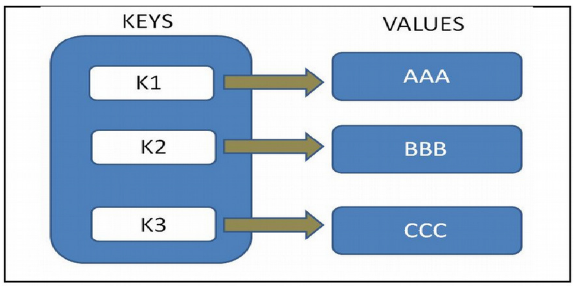

Fig. 1. Key Value Pair

Key Value Database is generally useful for warehousing, client reports, session data, shopping cart information, inclinations, etc. It should be avoided as soon as there is a necessity for data quering which have relations within the data i.e. stored actually or want to function at the same time over multiple keys.

Examples of Key Value Pair Database are: Redis, CouchDB, Riak, Memcached, Berkeley DB, Upscale DB, DynamoDB, Voldemort, etc.

B. *Document oriented Database*

A document store or document-oriented database is defined as a computer software package intended for warehousing, retrieving, and handling document-oriented data. It can be called as semi-structured data. The fame of the term 'document store database' has developed the utilization of idiom NoSQL procedure. XML documents are being adjusted to work with XML documents which are subclass of document-oriented databases. [2][4] Documents may comprise of collections, maps and scalar values which can be self-illustrating, categorized, tree data structures.

Document Databases can be useful for blogging programs, web analytics, content management techniques, real time analytics, and ecommerce systems. It should be avoided for applications that require composite transactions concerning several queries or operations in contradiction to variable aggregates in structures.

Examples of Document oriented Database are: MongoDB, CouchDB, Terrastore, OrientDB, RavenDB, etc.

C. *Column Family Database*

A NoSQL object that contains columns of related data is a column family. A key-value pair is a tuple (pair) where the key exemplifies a value which can be a set of columns. A column family is a table where each key-value pair can be represented in the row, like in relational databases. A column name, a value, and a timestamp are the tuples comprised in each column. Within a table along with other non related data, this data could be fabricated collectively, in a relational database table.[2][5]

Comparison of the container of rows in table where the key recognizes the row and the row comprises of multiple columns in every column family. A name and a value which can be used for mapping of the column are included in the Super Column which is the container of columns.

Column Family Database are convenient for blogging platforms, preserving counters, content management systems, terminating procedures, intensive write levels for example Log Aggregation. It should be avoided for systems that are in early development and have changing query patterns.

Examples of Column Family Database are: Cassandra, HBase, HyperTable, DynamoDB, etc.

D. *Graph Database*

A NoSQL database which uses graph structures for interpretation queries together with edges, nodes and properties to denote and warehouse the data, is known as graph database. The main hypothesis in this system is the graph along with relationship or the edge, which can precisely relate to the store data items. The relations permit store data to be correlated in sync unambiguously, to be retrieved in numerous cases with a single process.[2][6] Traversing the relationships and the joins are boosted by the Graph databases.

Graph databases are appropriate for problems which have data connected like social networks, routing data for money and goods, spatial statistics and for the recommendation engines.

Examples of Graph database are: Neo4j, Infinitegraph, OrientDB, FlockDB, etc.

TABLE I. COMPARISON OF VARIOUS NOSQL DATABASES

Name	API	Language	Concurrency	Replication	Misc.
KEY VALUE STORES					
Redis	Several Languages	C	Asynchronous saves in memory	Master/ Slave	Rich Set of Data types
CouchDB	HTTP	Erlang	Eventual Consistency(Availability and Partition Tolerance)	Selectable Replication Factor	Built for offline devices
COLUMN STORE					
Cassandra (Facebook, Twitter)	Many Thrift Languages	Java	Eventual Consistency(Availability and Partition Tolerance)	Multi-version Concurrency Control (MVCC)	Combination of Dynamo and BigTable
Hypertable (Rediff)	Thrift (Java, PHP, Perl, Python, Ruby, etc.)	C++/ HQL (Hypertable Query Language)	Strong Consistency (Consistency and Partition Tolerance)	Multi-version Concurrency Control (MVCC)	High performance with C++ implementation of Google's BigTable, Commercial support
DOCUMENT ORIENTED					
Couch DB	Representational State Transfer	Er-Lang/ JSON	Eventual Consistency(Availability and Partition Tolerance)	Multi-version Concurrency Control (MVCC)	JSON object queries, Better durability
MongoDB	Variety of dynamic object APIs available	BSON	Eventual Consistency(Availability and Partition Tolerance)	Master and Slave appr-oach	Query builder including a Javascript Map reduce implementation, GridFS specification

V. CASSANDRA

Apache Cassandra [7][8] is an open-source and a distributed database management system which can manage servers with data of huge amounts, delivering no single point of failure with high availability. Robust backing can be supported by Cassandra for clusters which comprise multiple datacenters.

To dominate the Facebook inbox search feature, Cassandra was invented at Facebook. On July 2008, Cassandra was launched by Facebook as an open-source project over Google code.

Cassandra is highly scalable, fault tolerant, consistent, and reliable, delivering high performance in distributed database. It has flexible data storage which can suit all possible data formats including structured, semi- structured and unstructured data. It has fast write speed without sacrificing the read efficiency. Users can access Cassandra via its nodes using Cassandra Query Language (CQL).

A. *Cassandra overall structure*

1) Data Model:

Table can be described using a key which is used to index multi-dimensional map. Column families are made by grouping columns. Simple and Super are the two types of Column Families. Every Column consists of Name, Value and Timestamp.

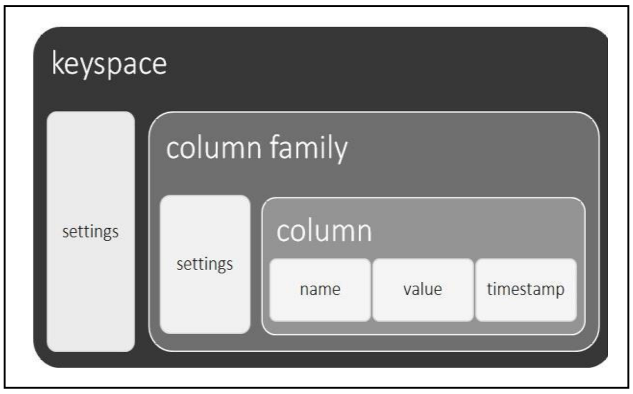

Fig. 2. Cassandra Data Model

1) System Architecture:

a) Partitioning: Ring topology is used for structuring nodes logically. The node in the ring is being assigned by hashed value of key related to the data partition. For backing the ring composition hashing rounds off after specific value. And to alleviate the highly loaded nodes it moves its position.

b) Replication: How the data is reproduced through the nodes is expressed using replication factor. At N (replication factor) nodes, each data item is being replicated.

i) Rack Unaware: The data at being replicated at N-1 successive nodes following its coordinator in this approach.

ii) Rack Aware: Zookeeper service is used which elects the leader to rectify nodes their range for replication in this approach.

iii) Datacenter Aware: This is analogous to Rack Aware however the leader is being elected at Datacenter level as a replacement for Rack level.

c) Cluster Management: Cluster Membership describes how nodes are added and deleted to the cluster nodes. It also describes how the communication between cluster nodes will take place.

Gossip Protocols are used for periodic, pairwise and inter node communication. Motivated from actual life rumours these are network communication protocols are being developed. Low cost factor is ensured by them delivering low frequency communication. Nodes are selected randomly for communication.

For e.g. – Node A requests to explore for a data pattern

Round 1 – Node A locally searches first, then gossips with node B.

Round 2 – Node A and B then gossip with C and D.

Round 3 – Nodes A, B, C and D then gossip with 4 other nodes and so on and so forth.

Round by round doubling makes protocol very robust.

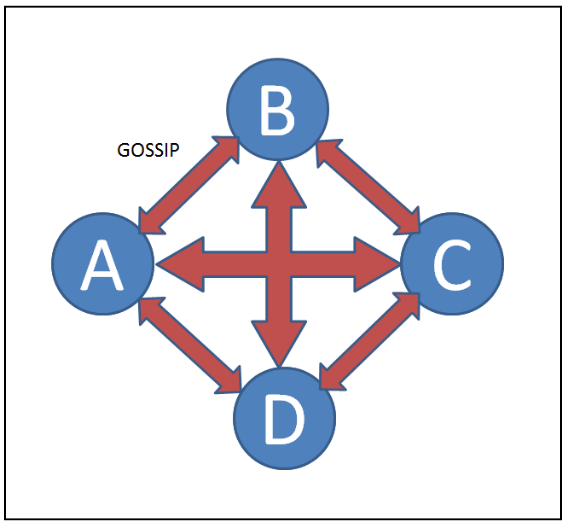

Fig. 3. Typical Gossip between nodes

Scuttle Back, a gossip protocol is used for managing the nodes in cluster. Variable '*phi*' provides distinctive node fail state which expresses by what means a node could fail (suspicion level) instead of the simple binary values (up/down).

There are two ways to add new node:

i) A random token is being allotted to the new node which provides its ring position. To the rest of the ring its location is gossiped.

ii) New ring node reads its configuration file to contact its initial contact point.

New nodes are included manually by administrator via CLI or Web interface provided by Cassandra.

VI. CASSANDRA PERFORMANCE TESTING

The test environment is as below. A Linux machine illustrates Cassandra representation. Four machines are used to fabricate the Cassandra cluster.

1) The Column Family Test:

The objective behind this experiment is to inspect and understand the relativity of the column families and their performance. In this experiment, 1000 queries are fired on every column family. Results are recorded in Table II and III.

The result demonstrates that large memory column family is backed by Cassandra, where speed of writing is considerably enhanced compared to speed of reading. Processing is slow for the input provided in case of single node for single database. The processing of provided input improves significantly in the multi-node environment for the same single database in distributed environment.

TABLE II. SINGLE SERVER SCENARIO

	Single Server Scenario				
No. of queries	1	10	100	500	1000
Reads/sec	10	94	156	147	170
Writes/sec	18	189	344	420	596

TABLE III. MULTI SERVER SCENARIO

	Four Server Scenario				
No. of queries	1	10	100	500	1000
Reads/sec	25	111	139	143	155
Writes/sec	6	29	170	368	489

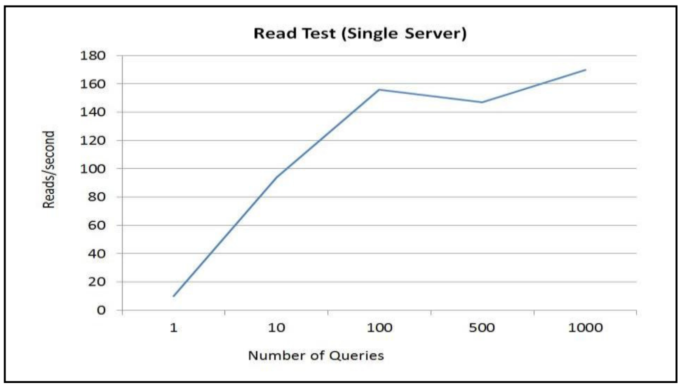

Fig. 4. Graphical representation for Cassandra Read Test in Single Server Scenario

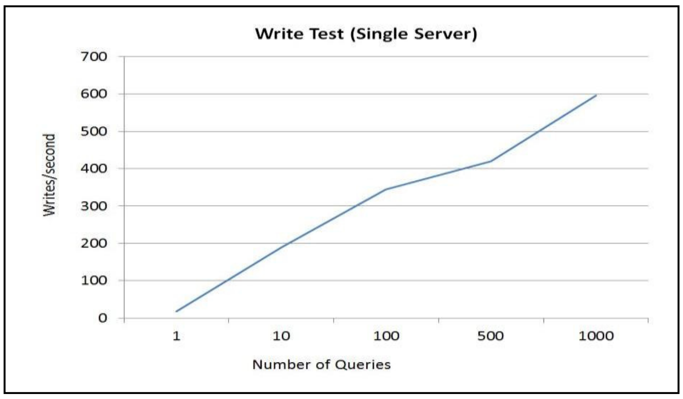

Fig. 5. Graphical representation for Cassandra Write Test in Single Server Scenario

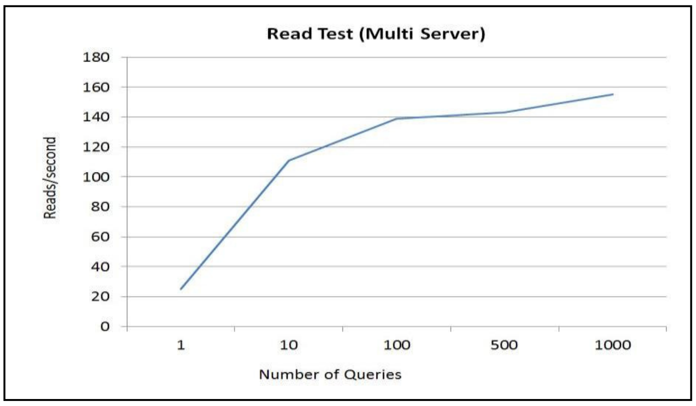

Fig. 6. Graphical representation for Cassandra Write Test in Multi Server Scenario

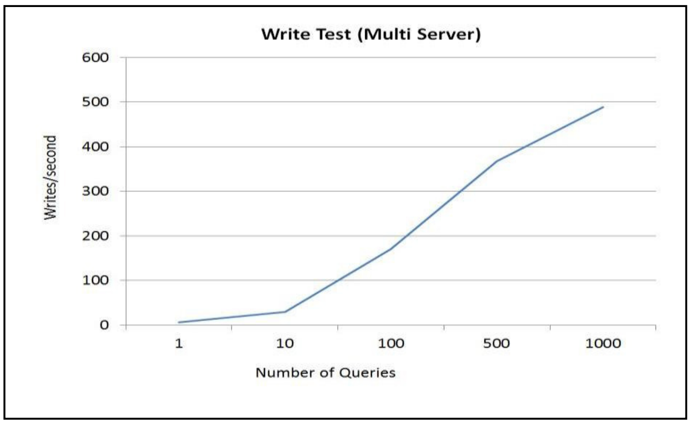

Fig. 7. Graphical representation for Cassandra Read Test in Multi Server Scenario

As per the graphs shown in Fig. 4 and Fig. 6, by using Cassandra in multi-node environment over single node environment, the timing to retrieve the outcomes from the queries are less than any other existing NoSQL databases. In this case only four servers are used.

In the case of write also as shown in Fig. 5 and Fig. 7, the graph shows that Cassandra surpasses other databases for insert operation in the case of multi-node for the identical single database over distributed environment.

VI. Conclusion

As the data size in internet is increasing extensively, NoSQL databases are used thus confirming to be a substitute for the traditional relational databases to a definite limit. The test result show that Cassandra improves its performance as the number of cluster increases with increasing access speed.

Many applications today use both relational and NoSQL database in combined format for resolving the problems in both the approaches.

References

[1] http://nosql-database.org/ [Online]

[2] Pragati Prakash Srivastava, Saumya Goyal, Anil Kumar, "Analysis of Various NoSql Database", *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*

[3] https://en.wikipedia.org/wiki/Key-value_database [Online]

[4] https://en.wikipedia.org/wiki/Document-oriented_database [Online]