Analysis of Various NoSql Database

Pragati Prakash Srivastava; Saumya Goyal; Anil Kumar (SMIEEE)
Dept. of Computer Science and Engineering
Mody University of Science and Technology
Rajasthan, India
prakashpragati@gmail.com, saumya.iis@gmail.com, dahiyaanil@yahoo.com

***Abstract*— In the age of internet, when data production has gone off-bounds, organizations are facing a tough challenge in terms of processing, analyzing and storing big data. The major drawback with this data is that it is not only being created at a lightning fast pace but it is also unstructured i.e. does not have a fixed schema. Moreover it is arising from disparate and discrete sources such as the social media. NoSql or Not Only Sql databases offer a highly flexible and horizontally scalable solution to store structured, semi-structured and unstructured data. These databases store data in the form of key-value pairs which offers better availability and high throughput performance in terms of processing queries. They are designed to be highly customizable according to the user's requirements, and well suited for the needs of the overlying application as well as the underlying data being stored. This paper firstly provides a general overview of the NoSql storage technology. Later a thorough analysis will highlight the features, strengths and limitations of six most popular NoSql databases and thus would help the organizations to choose a NoSql database which is well-suited to their needs.**

Keywords—NoSql, Big Data, Availability, Scalability, Consistency

I. INTRODUCTION

The web 2.0 era marked the beginning of a new age in data production. The web applications developed using relational databases started generating a new type of data (user data) which was unstructured in format, and was being produced at a very high rate, and belonged to disparate and discrete sources. This data was called as big data which was categorized by three V's (volume, velocity and variety) according to Gartner's definition [14]. Earlier the traditional databases would store all the application data in the form of structured record-based flat files which caused impedance mismatch and an arm struggle between application and database. This occurred since the applications were coded in non-declarative languages whose structure was quite different from these databases [1]. The phrase "one size fits all" is not applicable to the data being produced today. Traditionally we have always opted for vertical scalability-adding more components and resources to our system to increase its performance. However, since storage was not independent of application, it would cause service disruption and application reconfiguration each time a component was added. Most of the data we produce now is heterogeneous and therefore the underlying storage technology should not only be flexible to allow the data to be stored in its natural form but also be able to meet the cutting edge demands. NoSql databases or "key-value" databases offer a highly flexible horizontally scalable and fault tolerant solution to store structured, semi-structured and unstructured data [2]. They make use of commodity systems to scale-out the size of clusters and offer better availability and high throughput performance. Most importantly NoSql databases are designed keeping in mind the data that they need to store. This paper provides the readers a general overview of the big data storage types in section II. Section III describes the architectural working, strengths and weaknesses of six most popular NoSql solutions. Section IV summarises the results of the findings in section III. Section V discusses about future scope of research in this technology.

II. CATEGORIZING THE TYPES OF NOSQL

A. The Key-Value Data stores

These data bases will map a key to a value, or a set of values. Keys are unique and atomic which means they are indestructible. They are used to query the entries in the key-value storage databases. Key-value storage provides hash table implementation with key-value pairs spread across multiple remote servers in a distributive cluster [2]. Thus they can achieve the required efficiency by providing extremely fast random read/write requests and the flexibility to store data in a schema-less format. Since different key value pairs store a set of unrelated data it avoids the SQL join and GROUP BY operations as well as foreign key references. Key value stores are used by Twitter to store tweets with a unique twitter id. The corresponding values may include actual message, identity of the user and time of his post. Figure 1 demonstrates a key-value storage system.

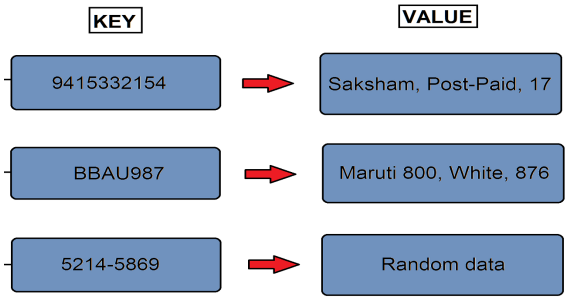

Fig.1 Key-Value Storage System

B. The Column Familty Database

The idea behind relational storage databases was that they would store all table entries associated with a single row together on disk, i.e. all the column entries associated with a particular row id would be stored together. In case of banking or financing organizations that had to maintain a huge set of related records, it was not always guaranteed that all values would be stored in a consecutive fashion. In case of the column family database an entire column of a table is stored together and is mapped to a single key. Since all the entries in columns have indexes, it is possible to search in only a part of the table. Also a column can have hierarchies of nested columns inside it making it the super column [2]. This provides easy look-ups and fast access while avoiding unnecessary overheads to look for individual key of a record. Figure 2 illustrates how data is stored in a column family database.

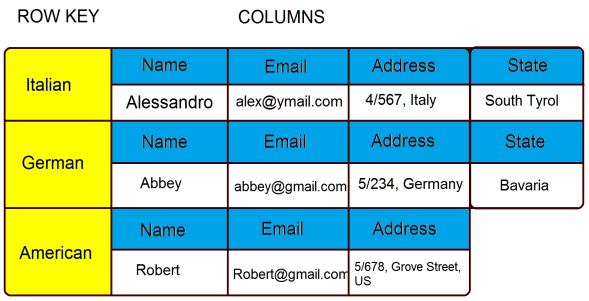

Fig. 2 Column Family Database

C. The Document Store Database

The document database is an alternative to the relational databases and is used to store semi-structured data existing in the form of XML (Extensible Markup Language), JSON (Java Script Object Notation), or in other similar formats [3]. A document can be compared to a row of a relational database which contains all related information about documents. A *collection* comprises of multiple documents where each document can have different schemas and differ on the number and type of data being stored.

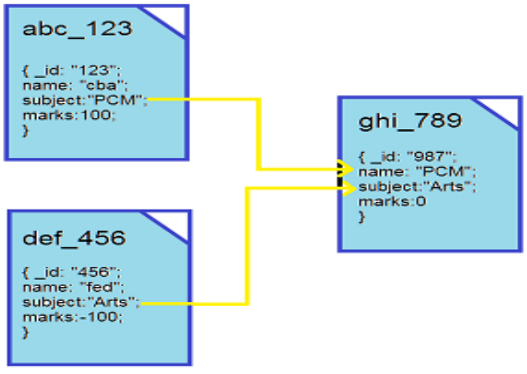

Fig. 3 Document Store Database

It is especially optimized to store textual information. Since related data set is stored together it saves the overheads of SQL JOIN operation [2]. Although the database has a schema free design, the stored records are semi-structured and exist in the form of hierarchies. Figure 3 illustrates an example of a document store database.

D. Graph Database

Graph databases are best suited for traversing and searching applications, such as finding related links on LinkedIn, looking up friends on Facebook etc. It gives more importance to the relationship between data items rather than data itself. They are highly optimized for fast traversal and make efficient use of the graph algorithms such as shortest path first in order to find the link between information. Figure 4 provides an example of traversing in a graph database.

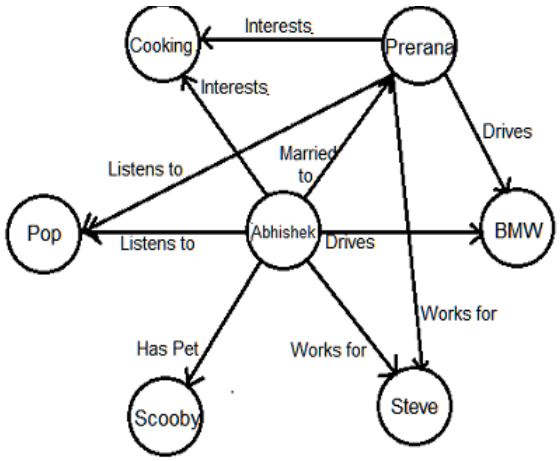

Fig. 4 Graph Database

III. UNDERSTANDING REQUIREMENTS BASED ON CAP THEOREM

Since data is highly customized and no two organizations produce same data it becomes important to firstly determine our expectations from NoSql database. For this a basic understanding of CAP theorem is required.

According to the CAP theorem it is impossible for a distributive system to simultaneously achieve Consistency, Availability and Partition Tolerance [2]. One of the three tenets must be sacrificed in order to achieve the other two.

- CA: All nodes in a cluster are constantly communicating with each other to ensure a strong consistency after each read operation. The stored data is always available to serve client's request however when partitioning or horizontal scaling is performed, the cluster gets blocked.
- CP: This system offers strong consistency of data after every transaction while also providing scalability without disrupting the application. However some data may not be accessible to serve client's request every time.

- AP: This system guarantees availability of data for serving client's request and high partition tolerance but offers eventual-consistency i.e. after completion of a transaction all replica's of the data might not be in the same state. However given no inputs all replicas will eventually become consistent across the cluster.

A. The Consistenent and Available Database

These principles are achieved by relational databases having ACID (Atomic, Consistent, Isolated and Durable) characteristics.

B. The Consistent and Partition Tolerant Data Storages

These systems have master-slave architecture which compromises on availability since all slaves will be useless without master node directing them. Under this category the most popular databases are:

- MongoDB
- HBase
- Redis

MongoDB is a document store database which stores semi-structured data written in JSON or JSON like language into BSON (Binary JSON) format [15]. It offers strong consistency since after a write commits successfully to memory the read operation will deliver updated results. It is also highly scalable. However it may offer limited availability of data since it is has only the primary master in a cluster. Thus in the event of a network failure the data might not be available. It can trade-off between availability and consistency by having more than one active masters in a cluster. This database was developed by 10gen and offers dynamic schemas- supports mapping of documents using both embedded and referenced approaches [2, 3]. 10gen offers both the free version under public licence type and enterprise edition under commercial licence. Some salient features of MongoDB include:

- It provides indexing of every attribute in a document and therefore delivers high performance.
- It offers automatic sharding mirroring and load-balancing of data across nodes and can scale-out without disrupting application [3].
- It uses GridFS file system for automatic sharding and storing larger files specially the unstructured multimedia data-e.g. videos. Provides special support for location and range based queries using latitude, longitude.
- It supports capped collections which are similar to circular buffers and offers high throughput performance [4].

MongoDB is best suited for the following:

- As an alternative to web applications using RDBMS.
- For providing high scalability and caching operations.
- For content management of semi-structured data.

MongoDB is not suited for the following:

- Applications requiring excessive JOIN operations and foreign key referencing.
- Applications requiring high conformity to ACID properties.

HBase is Apache's open-source hybrid column-oriented database which is an imitation of Google's BigTable [3]. It is mounted on top of the HDFS file system and can be useful in accessing only a part of the entire data set by using keys. A table in Hbase comprises of *regions*- storing a range of rows across distant Regional Servers. One major enhancement in the HBase's architecture over MapReduce is the division of the name node's task between HMaster and Zookeeper. While the task of Zookeeper is to interact with clients, the HMaster coordinates the work between different data nodes. It is horizontally scalable but can experience downtime when adding new nodes in cluster. It gives a little structure to the otherwise unstructured big data stored in HDFS. Some features of HBase are:

- It supports auto-sharding and replication data across nodes.
- It provides automatic load balancing and fault tolerance.
- It makes use of Bloom Filters (used to test whether an element is a member of a set) for real time querying [3].

HBase is best suited for the following:

- Applications involving a large number of random reads/write to BigData [3]. E.g. it is used by Facebook for retrieving photos, posts.
- Suitable for performing range-based queries.
- Applications requiring consistency in operations.

HBase should not be suited for the following:

- Typical OLTP and relational analytics. It is an alternative and not a replacement of RDBMS.
- Applications requiring scanning of extremely large datasets. Hadoop is better optimised for such tasks.

Redis is highly evolved open-source, BSD licensed, key-value storage system developed by Salvatore Sanfilippot [5]. It is one of the fastest in-memory databases. This database offers lightning-fast speed to support high read/write on workloads [5]. Unlike other NoSql databases Redis is also called as the data-structure server since it offers a great diversity of data structures in storing keys [3]. The keys can contain Strings (int, long, char array, linked list), Hashes (long int, double, char array), lists, sets, and sorted sets. Some features of Redis include:

- While storing data into the server users can specify time after which that set of record will be automatically discarded.
- Since it is an in-memory database it supports concurrent locking and writing of data and supports batch processing of data by using commands like EXEC, MULTI, and WATCH. Once all tasks are completed, the entire transaction is appended to the disk [6].
- The binary protocol it uses for communicating with other nodes is highly efficient; however its value is limited by a size of 512 Mbytes [3].
- A Single commodity system with 2.5 GHZ RAM can perform 1000-4000 writes/second in Redis.

Redis is recommended best suited for the following:

- For performing fast real time analytics- Redis provides the advance analytics feature of viewing the exact count of users using a website in less than a second [7].
- For content distribution and management, caching and cookie storage.
- For applications involving very high number of writes it will deliver high performance however the size of dataset cannot be larger than the size of main memory.

Redis is not recommended best suited for the following:

- Highly secure transactional operations on dataset since untrusted clients can also access the TCP/ Unix socket [9].
- Redis compromise on data availability since in case of a network failure if the address (port number) of master node changes a manual intervention is needed and setup has to be reconfigured [10].

C. *The Available and Partition Toletant NoSql Databases.*

These databases follow a peer to peer architecture unlike the classical master-slave architecture. Therefore they offer high availability at the cost of consistency. Under this domain the most popular data stores are:

- Cassandra
- CouchDB
- DynamoDB

Cassandra is a fully distributive open-source column oriented database developed by Apache [17]. It follows a ring architecture where all nodes are independent equal. It ensures partition tolerance and availability offering no single point of failure. Cassandra provides automatic load balancing and stores data in a denormalized form. It is a classical example of BASE (Basically Available, Soft state, Eventually Consistent) systems [16]. It supports shared nothing architecture since the client interacts with a proxy server and has no idea of where his data is being stored. Cassandra automatically divides the key-space and distributes it across nodes in a Cassandra cluster. The following are some characteristics traits of Cassandra:

- It offers linear scalability no matter how big the workload is-its throughput performance remains unaltered while crunching extremely large set of data.(The performance of other NoSql databases such HBase and MongoDB may degrade when processing a such huge amounts of data)
- It can trades-off between consistency with an increase in write latency. Cassandra uses gossip protocol to communicate an update message to all replicas simultaneously.
- Reading, writing, updating is extremely simple in Cassandra using built in queries. It offers a good user-experience and is almost perfectly fault tolerant.

Cassandra should be selected for the following:

- An application where number of reads are more than the number of writes. E.g. organizations like Airship, Twitter use Cassandra.
- Applications where immediate consistency is not a major concern. (In case of online hotel and flight booking users are interested in viewing only the lowest price, which may alter slightly at the time of booking.)
- Web applications that have to provide dynamic schema and content to users, e.g. Netflix.
- Applications were high maintenance of code is required.

Cassandra should not be selected for the following:

- For transactional and relational operations where high consistency is required.
- For dynamic querying involving JOIN and Aggregate operations.

CouchDB is a document oriented database written in Erlang and developed by Apache [11]. It focuses on the ease of use and completely embraces the web applications [11]. The data is stored in JSON JavaScript with MapReduce to perform queries. Couch supports self contained documents having different schemas. It supports multi-master bidirectional incremental replication and automatic fault detection [12]. The document metadata contains revision information which makes merging of disconnected documents extremely easy. Some salient features of CouchDB are:

- The built-in web administrator Futon directly interacts with database using HTTP protocol [3].
- CouchDB implements multi-version concurrency control which negates the need to lock data during write/update operations.

- It achieves document level ACID constraints with eventual consistency.
- For dynamic querying involving JOIN and Aggregate operations.

CouchDB can be used for the following applications:

- Applications involving periodic logging such as financing organizations where data is regularly modified and updated on the server.
- Employees of a construction company who have to send their workforce to inaccessible and remote locations can access and download the data using CouchDB.
- It can be an alternative to the MySql relational database in web applications offering better reliability and flexibility of design and scalability.

CouchDB is not a complete replacement of RDBMS since:

- Documents are stored in an isolated fashion and operations involving merging or aggregating two documents can be performed on the application layer and not on the database. This can be unfeasible for web developers at times.

DynamoDB is a key-value NoSql *service* provided by Amazon [18]. Users can conveniently store data in the form of a table comprising of items and attributes (similar to rows and column of Relational database). Since each item can have multiple attributes it allows flexible schemas to store data according to user's needs. Each attribute can be queried using a unique *Hash* key which acts like the primary key. The users can also combined Hash and Range keys to together act like primary keys. One distinguishing feature of Dynamo is that it completely automates the task of database administrator. It takes care of issues of scalability, provisioning, load-balancing, reliability, elastic MapReduce integration and ensures synchronous replication to ensure no data is ever lost. It is like a cloud based SAAS (software as a service offering) where users have to pay for throughput provision and amount of storage that one demands Moreover users interact with DynamoDB using a rich web-server API. Some features of DynamoDB:

- The users can define their required performance in terms of the number of reads and writes per second. DynamoDB will provision services accordingly. Typically in DynamoDB a single write per seconds is equivalent using 1000 bytes of data per seconds.
- DynamoDB offers two kinds of reads-eventual consistency and strong consistency. By default the reads are eventually consistent; however the users can opt for strong consistency where an immediately updated value will visible after a write operation. Strong consistency causes more write latency and is slower (half the throughput) when compared to eventual consistency.
- When altering the level of throughput or provisioning there is no loss of data or application program disruption.
- Retrieving the stored data is much faster than other NoSql databases since DynamoDB stores it across Solid State Drives rather than hard disk drives.

DynamoDB can be deployed for the following applications:

- It should be used when simple Create, Update, Delete (CRUD) operations are to be performed over a huge data set. E.g. online gaming, click stream tracking and web site applications.
- It is used to store items in Amazon's cart during online shopping.

It is not suitable for the following:

- Applications involving relational JOIN operations, and normalizations of data from different tables.
- In applications where the number of reads and writes per second can drastically vary such that it exceeds the read/write provision limit.

IV. Results

In section III we presented a detailed description of the six databases and their strengths and weaknesses. Based on this study we have the following findings.

- If data exists in the form of hierarchies and is in XML format and high level of consistency in operations is required **MongoDB** would provide the best solution. Examples of such applications would include online ticket booking portal. However one must keep in mind that since it has master–slave architecture it can cause rare service disruption in case of a master failure.
- If an organization majorly produces unstructured data at a high velocity and high performance in terms of processing random read/write requests is required then **Redis** would be the best solution. However since it is an in-memory database the size of transaction cannot be greater than the size of main memory.
- Also if per transaction a high volume of data is required to be processed then **Cassandra** would be able to deliver consistent performance and would be linearly scalable. Applications include online hotel booking or flight booking. Users should however be aware that Cassandra cannot be used for applications involving a strong consistency.

V. Conclusion And Future Work

Currently the NoSql technology is emerging and a lot needs to be discovered. According to a study conducted by the Information Week magazine, 44% of organizations do not know what NoSql databases are [13]. It is important to realize the potential that this storage technology carries which can

cause a massive paradigm shift in an organization's method of storing and processing data. In this paper we have evaluated the most popular NoSql solutions and also discussed their architectural working and best use cases. In future an objective comparison of these databases using fixed workloads of reads and writes can be carried out to compare their relative performance.

VI. REFERENCES

[1] Daniela Florescu and Ghislain Fourny, "JSONiq: The History of a Query Language" , Internet Computing IEEE, vol. 17, Issue: 5, 2015, pp.86-90. (*references*)

[2] Jagdev Bhogal and Imran Choksi, "Handling Big Data using NoSQL" **"**IEEE Conference Publications on Advanced Information Networking and Applications Workshop (WAINA)", pp. 393 - 398, 24-27 March 2015. (*references*)

[3] Ganesh Chandra Deka, "Fine A Survey of Cloud Database Systems", "IT Professional" vol. 16, Issue: 2, 2014, pp. 50-57, 03 January 2013

[4] "Capped Collections" Internet: http://docs.mongodb.org/manual/core/capped-collections/, 2011 2015 [7/9/2015].

[5] Redis Labs "Redis FAQ" Internet: http://redis.io/topics/faq, [8/9/2015].

[6] Redis Labs "Redis Transactions" Internet: http://redis.io/topics/transactions , [8/9/2015].

[7] "PFCOUNT key [key ...] " http://redis.io/commands/pfcount , [8/8/2015]

[8] Redis Labs "Redis Security" , Internet: http://redis.io/topics/security , [8/9/2015]

[9] Redis Labs "Redis Security" Internet: http://redis.io/topics/security, [8/9/2015].

[10] "Guidelenes for redis Client with Support for redis Sentinal" Internet: http://redis.io/topics/sentinel-clients

[11] Apache Couch Labs. "A Database For The Web" , "http://couchdb.apache.org/ " [10/9/2015]

[12] Wikipedia contributors "Multi Master Replication" Internet: https://en.wikipedia.org/wiki/Multi-master_replication , 10 September 2015 , [10/9/215]

[13] Information Week magazine "Surprising: 44% of Business IT Pros Never Heard of NoSql" Internet: http://www.informationweek.com/database/surprise-44--of-business-it-pros-never-heard-of-nosql/d/d-id/1092523? [10/9/2015]

[14] Gartner Research "Big Data" http://www.gartner.com/it-glossary/big-data [10/9/15]

[15] MongoDB Labs "JSON and BSON"https://www.mongodb.com/json-and-bson. [10/9/2015]

[16] Charles Row "Acid vs BASE: Shifting pH of database Transaction Processing. " http://www.dataversity.net/acid-vs-base-the-shifting-ph-of-database-transaction-processing/ ", March 1, 201. [9/9/2015]

[17] Apache Labs , "Cassandra" ," http://cassandra.apache.org/ " [10/9/2015]

[18] Amazon Web Services "What is Amazon DynamoDB? " , "http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html" , [10/9/2015]