IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 6, Ver. IV (Nov.-Dec. 2016), PP 06-12
www.iosrjournals.org

The NoSQL Movement

Prof. (Dr) Mohammad Ubaidullah Bokhari[*], Afreen Khan[*]
[*]*(Department of Computer Science, Aligarh Muslim University, Aligarh, India)*

Abstract: *With the advent of technology, the growth and usage of web data has increased and hence the need to store such large amount of data is rising at an exponential rate. It has thereby caused an immediate need for a novel database management system that can deal with the storage of huge amount of unstructured and semi-structured data. The ever-increasing fame of Cloud Computing and Big Data is the stirring power for the companies to make their shift from classic relational databases to non-relational datastores. These non-relational databases are better referred as "NoSQL", meaning "Not Only SQL." NoSQL databases are not a replacement theory for the relational databases but rather they came into existence with the notion of dealing with the unstructured data very well. These databases are tremendously being used by the organizations because of their high performance and excellent scalability. As it deals with the huge amount of user-sensitive data, these datastores raise the alarm for data integrity, confidentiality, injection and insider attacks, consistency, and availability of data, and hence the security provided by these datastores. The objective of this paper is to summarize the research journey of NoSQL done up till. It presents NoSQL database with respect to its historical background, overview, basic concepts, pros and cons, limitations and its security loopholes. It later focuses on the six chief scenarios that are accountable for the threat model of NoSQL databases.*

Keywords: *Big data, NoSQL, Structured data, Security, Unstructured data*

I. Introduction

The coming of the Big Data Era is the major break for organizations and individuals in the field of technology since the crack of dawn of the Internet**.** If we stride back for an instant and take glance at how the technology world has transformed since the turn of century, we can see that world's unstructured information is about 80%, unstructured information is rising at 15 times the speed of structured information, unprocessed computational power is increasing at such a huge rate that today's off-the-shelf service box is opening to display the power that a supercomputer depicted half a decade back, and right of access to information has been democratized [1].With the sudden increase of computing power athwart midframes, mainframes, and personal desktops, datastores have turn into an omnipresent means of collecting and storing data. Actually, their development led to the formation of a division of knowledge roughly about warehousing the data, such that it was simpler to control and correlate data from numerous databases in a consistent fashion. Different portions of a company used diverse data management systems to stock up and search their significant data, by the late 1990s. It was in 2001, when the era of XML came up. Though XML approved a flexible schema and simplicity of portability as chief advantages, the extensive usage of accretion of back office content, e-mail, and other technologies led to the requirement for content management systems [1]. And then the age of analyzing semistructured and unstructured data in enterprises was born. At present, with the arrival of the Internet joint with entire democratization of content formation and sharing in numerous formats, has led to the bang of all kinds of data- in terms of volume, variety and velocity.

II. The NoSQL Evolution

Since the data is increasing at a rapid pace, an enormous need arise to knob and store the unstructured data. Traditional databases have resulted in intricacy that keeps generally companies spinning in loop. Organizations cannot keep up with the many sizes, shapes, and kinds of data that are rapidly growing in quantity and incessantly altering. They have to resist hard so as to tackle the ever rising unstructured data. Traditional databases can't classify unstructured data. These datastores are developed and prepared to house the structured data such as economic data. RDBMS are deficient in elevated velocity since it's designed for sturdy data maintenance rather than fast growth. If RDBMS is made to tackle and hoard big data, it will definitely revolve out to be costly. Consequently, the incapability of relational datastores to deal with the unstructured data led to the need of new methodologies. Thus, a new method is needed that help deal with such kind of data in a simple and proficient way. And this is the case, where NoSQL datastore comes into the picture that helps to tackle the unstructured data in a better way. NoSQL is not at all a movement against the typical relational databases. The "NoSQL" term means "Not Only SQL." NoSQL datastores are the probable way out for performance and scale. Web services produce huge quantity of data every hour; hoarding this large amount of data and managing the elevated performance is fetching challenges nowadays. This provides additional possibilities ahead of the

conventional approach of data perseverance to the software developers. The NoSQL points to a huge group of non-relational datastores that majorly vary from conventional RDBMS in various major aspects, mainly because they don't make use of SQL as their chief query language rather use APIs. The most significant question in evaluating the wants of an application and whether to utilize NoSQL engines or relational databases basically depends on the kind of application being written, the nature of queries that are accepted, and the constancy vs. unpredictability of the data's structure.

III. Historical Background

1. It was the era of mainframe processors that involved only one CPU for the processing and storage. It was then when languages such as COBOL and FORTRAN and punched cards and flat files for storage were used. The expenditure cost was about $10,000 per CPU hour. After that, commodity processors came into light that involved 10,000 CPU's for the processing and storage. Functional programming languages such as Erlang, F#, Wolfram etc are used. It employs MapReduce technique and the expenditure cost about pennies per CPU hour.
2. In the 1930s-40s period, John von Neumann and Alonzo Church proposed computation approaches. Neumann's computation way involved managing a state with a program counter while Alonzo's approach involved making computations act like math functions (Fig 1).

Figure 1: Standard vs. MapReduce Cost

3. MapReduce CPUs Cost Less:

The graph below (Fig 2) implies a direct cut cost from 16.9 to 3 cents per CPU hour in between EC2 and MapReduce. It entails that Alonzo Church was right for his theory of making computations act like math functions.

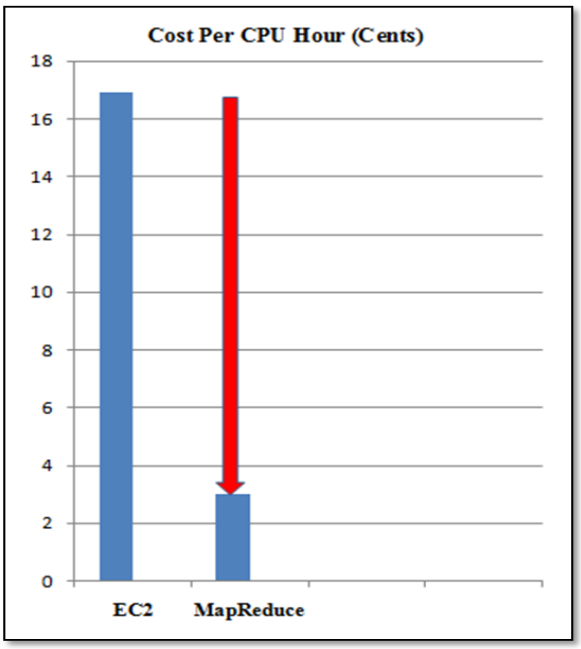

Figure 2: EC2 vs. MapReduce

IV. Motivation Behind NoSQL's Development

In 1998, NoSQL was used first in place of a relational database that omitted the utilization of SQL. This NoSQL term was chosen again in 2009 and it was used for "conferences of advocates of non-relational data stores" [3]. Eric Evans, a blogger, described the ultimate goal of the NoSQL faction as "the whole point of seeking alternatives is that you need to solve a problem that relational databases are a bad fit for" [4]. An engineer Avinash Lakshman reported that NoSQL is capable to write 2500 times quicker into a 50 gigabytes huge database than MySQL [5]. The Computerworld magazine's article sums up the reasons behind the development and usage of NoSQL datastores. It is as follows:

1. To avoid superfluous Complexity: Relational databases give a range of features and firm data consistency. But the ACID properties employed by RDBMSs may be more than essential for exacting applications and use cases.
2. High Throughput: Certain NoSQL databases offer a considerably elevated data throughput than conventional RDBMSs.
3. Horizontal Scalability: In distinction to RDBMSs, most NoSQL datastores are designed to scale fine in the horizontal route and not depend on vastly accessible hardware.
4. To avoid costly Object-Relational Mapping: NoSQL do not make high-priced object-relational mapping essential. It is usually vital for applications with data structures that possess low complexity that can barely advantage from the characteristics of a relational database.
5. Difficulty and expenditure of Setting up Database Clusters: NoSQL databases are developed in a manner that PC group can be simply and inexpensively expanded devoid of complication.
6. Negotiation of Reliability for improved Performance: There are many different cases where applications might be willing to negotiate reliability for enhanced performance. For example, HTTP session data can be shared among different web servers but as the data is temporary, there is no requirement to store it in unrelenting storage.

A comparison between worldwide usage of RDBMS and NoSQL as posed by Google Trends (from 13 Nov'2011 to 5 Nov'2016) is depicted below graphically (Fig 3).

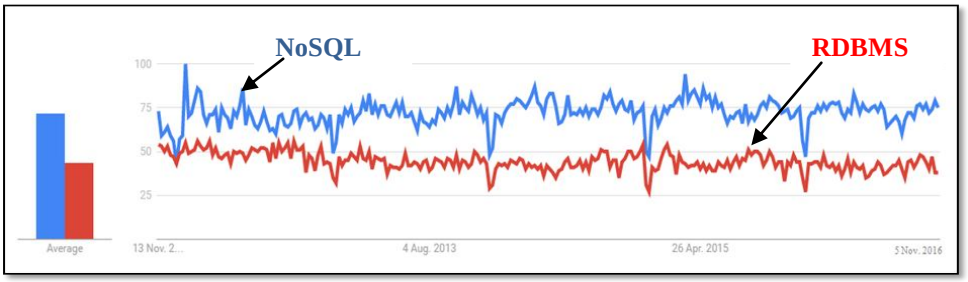

Figure 3: Google Trend for NoSQL

V. Basic Terms And Techniques

This section includes certain essential concepts, techniques and patterns that are prevalent among NoSQL datastores and are not only one of its kinds to any one group of non-relational databases or a sole NoSQL datastore.

1. The CAP-theorem: At ACM's symposium, Eric Brewer in the year 2000, came up with the notion of CAP theorem while addressing a keynote titled "Towards Robust Distributed Systems". The CAP short form stands for:

- **Consistency:** It reflects how a system remains in a consistent state after a certain operation is carried out.
- **Availability:** In this, a system is built and implemented in such a way which allows it to take out the many operations such as read and write.
- **Partition tolerance:** It means the ability of any system to extend operation in the presence of certain network partitions. A situation arises when two or more network nodes come up which, temporarily or permanently cannot be linked to each other. Then the partition tolerance serves the system to deal with the dynamic addition and elimination of nodes accordingly.

The main idea behind the CAP-theorem is that, it postulates that no more than two of the three different aspects of scaling out, can be completely achieved at the same time. Brewer also states that an individual can atmost choose two of these three characteristics in a shared data system.

2. ACID vs. BASE: The BASE (Basically Available, Soft state, Eventually consistent) characteristic as stated by Brewer relinquishes the ACID (Atomicity, Consistency, Isolation, Durability) features of consistency and isolation in account of availability, degradation, and performance. A contrast table between ACID and BASE is given in Table I below. Furthermore in his talk, he pointed out the trade-offs between the ACID and BASE systems and recommended as a decision model to choose one or the other for particular use-cases: if any system or part of that system compass to be consistent and partition tolerant, ACID characteristics are extremely required and if on the other hand, availability and partition tolerance are advantaged over consistency, then the resultant system can be classified by the BASE properties.

Table I: ACID vs. BASE

ACID	**BASE**
A- a transaction must completely execute or not at all **C-** takes database from one consistent state to another **I-** a transaction must be executed in isolation; no interference from other concurrent transactions **D-** the changes committed must persist; not be lost because of system failure	**BA-** system assure availability in terms of CAP-theorem **S-** without any input, the state of system may possibly change over time **E-** given that the system doesn't accept any input, the system will turn out to be consistent over time

3. Difference between Strict Consistency and Eventual Consistency: Strict consistency implies that either read or writes operations for a particular dataset have to be carried out on the same node or distributed transaction protocol assures the strict consistency. The CAP-theorem mentions that the strict consistency cannot be attained jointly with availability and partition tolerance. While eventual consistency is used to attain high availability. It means that the reader will see writes and as the time passes by, in a stable state, the system however will eventually return the very last written value. The users as a result, may perhaps face an inconsistent state of data as updates continue to evolvement.

4. Clustering: It is a way to divide the data that strive for transparency towards users which shouldn't observe communicating to a group of database servers in its place of a single server.

5. Sharding: It refers to the partition of the data in a way that the data which is requested and updated mutually, reside on the similar node and on the other hand, the load and storage volume is generally distributed between the servers. There is also a case when the data shards ('shard' means 'a small fraction of total') can be replicated. It may be replicated for certain reasons of reliability and load-balancing.

VI. Pros and Cons of NoSQL

NoSQL databases are extremely scalable, reliable; possess a straightforward data model and consists of highly bare query language. The most important advantage of NoSQL database is that this database can handle unstructured data very well. The unstructured data includes the word documents, audio, video, emails, or social network data. These databases scale very well on commodity hardware. NoSQL databases facilitate better performance; to allow faster performance, these databases usually don't stick to ACID restraints that are basically used in relational databases. NoSQL databases do possess faults and limitations. A general concern with these datastores is that there are many security loopholes such as, they lack encryption support for data records, have weak authentication methodology, encompass lack of client communication encryption, and are vulnerable to SQL injection or denial of service attacks. Some of the NoSQL implementations are trying their best to tackle such issues but their proposed solutions are not however production ready.

The NoSQL datastores are not ACID compliant likewise relational databases. This may lead to problems with application that involve immense accuracy. All this can be tackled but not without adding up additional overhead and complexity. To ensure ACID compliance, additional programming must be done. As soon as these additional rules are implemented, database performance probably takes a hit. This is mainly the reason why the developers have selected to drop ACID property from the NoSQL databases as well as it is a chief aspect which needs to be kept in mind whenever selecting a database for a meticulous application. Still in some cases, NoSQL systems lag behind the optimized RDBMS. This implies that in certain cases RDBMS still has advantage over their NoSQL counterpart.

VII. NoSQL Data Modeling Techniques

The NoSQL database is categorized into four models. Each model has its own benefit and downfall. In general, a common idea is that the developers need not have to map the traditional tabular data to a non-tabular object within their programming language.

1. Key-Value Stores: Key-value NoSQL datastores are the simplest datastores. They basically store particular item as alpha-numeric identifiers (keys) and linked values in simple standalone tables (which is referred to as-hash tables). These keys may be simple text strings or can be more complex lists while the value is a BLOB and can be any kind of data that the database stores, devoid of knowing what's within. The ultimate responsibility is of the application to comprehend what was stored. The data searches can only be performed against keys, not values. It is limited to exact matches. Key-value stores access through primary key and hence, they show better performance and scalability. These data stores are used in applications that involve smaller reads and writes on a frequent basis. They are used in applications that deal with session management, transactions relating to shopping cart, storage of configuration and user sensitive data for mobile applications. They are not suitable if one needs to update or query the database.
Examples: Amazon DynamoDB, Riak, Redis, Oracle NoSQL

2. Document Databases: Document databases handle and store documents. Standard data format such as XML, JSON or BSON are used to encode the documents. These databases fit well if certain application needs to store different attributes and large quantity of data. A sole column can contain several of such attributes, and the quantity and type of attributes recorded can differ from row to row. Data that is often queried is basically stored jointly in the same document rather than storing it in different tables. Moreover, unlike Key-value stores, both keys and values are completely searchable in document databases. These databases provide flexibility for the data. They provide management of data types with inconsistent attributes and support websites that involve huge volumes of reads and writes. They are well-suited for applications that employ JSON data structures.
Examples: CouchDB, MongoDB

3. Column-Family Stores: Similar to Document datastores, Column-Family stores consists of distributed, column-oriented data organizations that accommodate numerous attributes per key. They are designed for storage of large quantity of data, high availability and read and write performance. These datastores execute on groups of multiple servers. In case, if the data is not large enough and can execute on a single server, then key-value store or document database must be used instead. Column-family stores are suitable for the applications that are distributed over numerous data centers. They are used in the field of bioinformatics using proteomic and genetic data. The applications that involve dynamic fields are queried well through Column-Family stores.
Examples: Google BigTable, Cassandra

4. Graph Databases: Graph databases are the databases that replace traditional database tables with structured relational graphs of interrelated key-value pairings. They are used when there is certain kind of connection or some direct relationship among two entities. They are depicted as an object-oriented network of nodes (objects), relationships (edges), and properties (object attributes articulated as key-value pairs). The Graph databases are the only NoSQL datastores that are concerned with relations, and they focus on visual demonstration of information. These are useful when one is concerned with relationships among data than in the data itself. The domains that involve representation as networks of linked entities are very well suited for graph datastores. It is used in demonstrating and traversing social networks, building recommendations, process management in businesses, performing forensic investigations.
Examples: Neo4j, InfoGrid, InfiniteGraph

Key-value stores, document databases and column-family stores are suitable for extensive range of applications. While graph databases are a perfect fit to a particular type of problem.

VIII. Reasons To Use NoSQL Database

As NoSQL is a fairly new technology, it has already fascinated a significant quantity of attention due to its utilization by enormous websites like Amazon, Facebook, Yahoo, which have data consumption rates that fetch relational databases to lag behind. Many products self-identify as NoSQL, and all possess their own unique structural design. The NoSQL solutions are very much attractive. They can deal with huge quantities of data quite speedily across a bunch of commodity servers that share many resources. In addition, most of the NoSQL solutions are open source. There are some important reasons that need to be considered for the use of NoSQL databases:

- To advance programmer productivity by using database that matches an application's requirements [6].
- To advance data access performance via certain grouping of handling big data volumes, lessening latency, and thereby, improving throughput [6].

IX. NoSQL Security Loopholes

The different faces of security and protection in a NoSQL environment are impending under the much greater scrutiny, as more and more data moves into and throughout the NoSQL systems and many more of those systems are generally deployed with a public-facing module (or even when they're not). It is not a very simple portrait. Here are six key NoSQL security problems as acknowledged by Cloud Security Alliance [2] that need to be focused upon:

1. Transactional Integrity: The most important drawback of NoSQL datastore is its elastic way towards ensuring transactional integrity. If multifaceted integrity constraints are introduced into the architecture, it will surely fail NoSQL's main purpose of acquiring better scalability and performance.

2. Loose Authentication Techniques: NoSQL databases make use of weak authentication techniques and feeble password storage mechanisms and thus, exposed to password brute force attacks and replay attacks, resulting in information leakage.

3. Ineffective Authorization Techniques: The authorization mechanisms vary from one NoSQL's solution to another. The important aspect is that there is no RBAC mechanism created into the design because defining the roles of user and security groups with an RBAC technique is not possible.

4. Susceptibility to Injection Attacks: Injection techniques allow backdoor entrance to the file system for vicious activities. As NoSQL design employs frivolous protocols and techniques that are insecurely coupled, it is more susceptible to a variety of injection attacks such as, JSON injection, view injection, array injection, REST injection, etc.

5. Lack of Consistency: It's the incapability to concurrently implement all three fundamentals of the CAP-theorem. Consequently, users are not assured consistent output at any known time, because every participating joint may not be completely coordinated with the joint holding the most recent image.

6. Insider Attacks: Light security techniques usually leverage to attain insider attacks. Such attacks could stay ignored because of meager logging and log analysis mechanisms, all along with the erstwhile basic security technique.

X. Discussion And Future Scope

Relational databases since years have been dominating the organizations. With the dawn of expertise, usage, growth and need of data, NoSQL databases are gaining more interest by the developers. The reasons for the same are:

- Big data and big data users require dynamic, flexible and schema-less data model. Hence, NoSQL datastore fulfils the above requirement of big data and its users.
- NoSQL databases have a capability to scale up radically so as to support global big data and its users.
- NoSQL databases provide an enhanced performance without negotiating scalability so as to satisfy big data users' expectation.
- Security methodologies must be put into practice at the middleware layer by the developers to conquer the security concerns of NoSQL datastores.

In contrast with the traditional databases, we need to make stronger databases without compromising the scalability and performance characteristics. NoSQL has an enormous growth in the future since most of the current applications and software are shifting to web and also the size of data needed to store is rising speedily. This implies that it will face development, improvement and enormous growth and, sooner or later it will surely solve its security problems.

This paper presented the journey of NoSQL. Furthermore, it analyzed the security loopholes and the threats present in various NoSQL databases. Certain objectives that need to be focused with respect to the security of NoSQL databases are:

- To deal with crucial security flaws in NoSQL databases
- To make official the threat representation for security of NoSQL and to discover easy manageable results based on the threat model
- To create and develop suitable security method for securing NoSQL datastores
- To discover security challenges that are expected in future so as to secure NoSQL databases

Acknowledgment

Prof. (Dr.) Mohammad Ubaidullah Bokhari is a Chairman at Department of Computer Science, Aligarh Muslim University, Aligarh, UP, India. Afreen Khan is a student of MCA at Department of Computer Science, Aligarh Muslim University, Aligarh, UP, India.

References

[1]. Chris Eaton et al, Understanding Big Data- Analytics for Enterprise Class Hadoop and Streaming Data, (USA: McGraw Hill, 2012) pp. xv-xviii.
[2]. Sreeranga Rajan et al, Expanded Top Ten Big Data Security and Privacy Challenges, Cloud Security Alliance, April 2013.
[3]. Evans and Eric, NOSQL 2009, Blog Post May 2009, http://blog.sym-link.com/2009/05/12/nosql_2009.html
[4]. Evans and Eric, NoSQL: What's in a name, Blog Post October 2009, http://www.deadcafe. org/2009/10/30/nosql_ whats_in_a_name.html
[5]. Avinash Lakshman and Prashant Malik, Cassandra- Structured Storage System over a P2P Network, June 2009, Presentation at NoSQL meet-up in San Francisco on 2009-06-11, http://static.last.fm/johan/nosql-20090611/cassandra_nosql.pdf
[6]. Website https://www.thoughtworks.com/insights/blog/nosql-databases-overview