Handling with Dynamic, Large Data Sets – NoSQL a Buzzword or Savior?

Tushar Maroo
(B.Tech. Scholar, JECRC Foundation, Jaipur, India, tusharmaroo@gmail.com)

Abstract

Researchers in Database vertical mention a number of topics that challenge current data storage concepts. Such as managing and creating collections of unstructured data, horizontal scaling, mobile and cloud computing. Relational databases discovered some limitations due to their underlying relational model; this article shows that newly developed NoSQL databases have characteristics meeting those challenges. NoSQL (Not only SQL) is a database used to store large amounts of data. NoSQL databases are distributed, non-relational, open source and are horizontally scalable (in linear way). NoSQL does not follow property of ACID as we follow in SQL. They are widely used in Web 2.0 applications that manage great, unstructured data collections with dynamic content. An example from Social Networking sites like Facebook dealing with the large amount of datasets underpins the argument that NoSQL databases should be considered as context data storage. Facebook reports that it has currently 500 million active users, 200 million of which access its services on mobile systems.[1] Context aware architectures intend to explore this increasing number of context information sources and provide richer, targeted services to end-users, while also taking into account arising privacy issues. Study is carried out about the characteristics, and pros and cons of NoSQL.

I. Introduction

NoSQL stands for Not Only SQL. It is one of the data storage other than databases (that were used earlier) that is used to store huge amount of data storage like data in facebook (which keeps on increasing day by day). NoSQL is a non-relational database management system (sometimes called as derived from relational database), fast information retrieval database and is portable. NoSQL basically derives from RDB database system. This database usually interacts with the UNIX operating system. NoSQL databases are those databases that are non-relational, open source, distributed in nature as well as it is having high performance in a linear way that is horizontally scalable. Non-relational database does not organize its data in related tables (i.e., data is stored in a non-normalized way). NoSQL databases are open source; therefore, everyone can look into its code freely, update it according to his needs and compile it. Distributed means data is spread to different machines and is managed by different machines so here it uses the concept of data replication.
With the advent of social networking sites like facebook and twitter, the demand of new technology that can handle huge amounts of data has lead the emergence of various new technologies and one of the prominent is NoSQL which is quite helpful in data warehousing. NoSQL (non-relational) is comparatively faster than relational databases. Previously, in SQL, we were using Query language to fetch as well as to store data; for NoSQL we store large data entities using documents in XML (eXtensible Markup Language) formats. XML language is basically used to store structured data in a human readable form. This is further discussed in later sections.

II. Characteristics of NoSQL

- NoSQL does not use the relational data model thus does not use SQL language.
- NoSQL stores large volume of data.
- In distributed environment (spread data to different machines), we use NoSQL without any inconsistency.
- If any faults or failures exist in any machine, then in this there will be no discontinuation of any work.
- NoSQL is open source database, i.e. its source code is available to everyone and is free to use it without any overheads.
- NoSQL allows data to store in any record that is it is not having any fixed schema.
- NoSQL does not use concept of ACID properties.
- NoSQL is horizontally scalable leading to high performance in a linear way.
- It is having more flexible structure.

III. NoSQL data store types

On the basis of CAP theorem[2] NoSQL databases are divided into number of databases. There are four new different types of data stores in NoSQL.

Key value databases: The key value databases name, itself states that it is a combination of two things that is key and a value. It is one of the low profile (traditional) database systems. Key Value (KV) databases are mother of all the databases of NoSQL.

- Key is a unique identifier to a particular data entry. Key should not be repeated if one used that it is not duplicate in nature.
- Value is a kind of data that is pointed by a key.
- It handles huge data load.
- It scales to large volume of data.
- Replication of data is done using database in the form of ring. The replicated data is stored in the form of ring as well as in the alphabetical order.

2. **Document Stores Databases**: Document Stores databases are those NoSQL databases which use records as documents. This type of database store unstructured (text) or semi-structured (XML) documents which are usually hierarchal in nature. Here each document consists of a set of keys and values which are almost same as there in the Key Value databases. Each database residing in the document stores points to its fields using pointers as it uses the technique of hashing. Document Stores Databases are schema free and are not fixed in nature.

- Documents are addressed in the database using key (unique) that represents that document.
- There are number of varieties to organize data that is collections, tags, non-visible metadata and hierarchies.
- In this we can use a key-value lookup to retrieve a document.

3. **Columnar Databases**: Columnar Databases are also known as column family databases because they are column-oriented databases.[3]

- Columnar databases are faster than row based databases while querying.
- In columnar databases, assignment of storage unit is done to each and every column.
- In the columnar DBMS only

the required columns are read, so reading is faster in this case.

4. **Graph databases**: Graph databases are based on the graph theory. In general, we see that graph usually consists of nodes, properties and edges.

NoSQL Graph database consists of:

1. Nodes represent entities
2. Properties represent attributes
3. Edges represent relationships

Graph traversals are executed with constant speed independent of total size of the graph. There are no set operations involved that decrease performance as seen with join operations in RDBMS.

- Graph databases are having high performance in context to their deep traversals.
- These are used for shortest path calculations.
- These are scalable. But its complexity increases.

IV. Context storage using NoSQL

Although in a Publish-Subscribe system[4] it is not necessary to store every published item, in a Context Management Platform it becomes advantageous to do so. Storing every piece of context information provides a comprehensive history of a user's context data, which allows for very powerful features such as context-aware advertising, actuation and environment adaptation according to the user's preferences. With the need to store every user's full context history come the problem of handling very large quantities of data. This is a very active research problem, with several proposed solutions such as the so-called NoSQL storage systems. The information to be published is in XML format. This information can have any structure, and the Platform should be able to efficiently handle context information with any XML structure. The type of queries used in a Publish-Subscribe system are simple listings of context information, grouped by published Node (E.g. the last 10 Items published on node X) and sorted by publishing date. It should also be possible to retrieve published Items by ID. This paper, then, proposes the usage of a NoSQL storage system for storing context information with the following main goals in mind: better performance when handling massive amounts of data, horizontal scalability and availability, and integration with a full-text searching engine. We begin by looking at performance advantages.

A. Performance Advantages:

As the XMPP Publish-Subscribe protocol is a good fit for centralized context management platform. CP's, which receive context information from CA's, can publish the context information in the CB, through XMPP PubSub Items. CCs, which subscribe certain context types (through XMPP PubSub Nodes) receives notifications with the context information, as published by the CPs. Besides real-time notifications from context publication, CCs can ask the CB for published context data. Due to the organization of the context data through XMPP PubSubNodes, it can retrieve, for example, the last Location Item published, or the last 10 Social Preferences Items. This makes for two distinct operations, with regards to context publishing. One will be a real-time notification to all the subscribed entities of a context type (PubSub Node), while the other corresponds to a request of context information. For the first operation, context information storage is largely irrelevant for the completion of the notification procedure. It is theoretically possible,

for example, to first notify all subscribed entities, and postpone the context data persistence to disk (due to the second operation, it may not be desirable). For the second operation, however, context data will be retrieved from storage. Due to the nature of the context information, it may be acceptable for the information retrieved to miss one context item that's being published in the exact moment the retrieve operation is executed. E.g. when publishing GPS coordinates every 10 seconds, it may be acceptable to, when retrieving the last published GPS coordinates, return the outdated GPS context data (although it will only be, at most, 10 seconds old; hardly meaningfully outdated). This relaxed persistency constrains allows the tradeoff of strong-consistency features, as present in relational databases, for weaker forms of consistency, when accompanied by performance, reliability and availability gains, A NoSQL storage system will do such a trade.

B. **Horizontal Scalability and Availability:**
Horizontal scalability refers to the ability of scaling a database by adding more machines or nodes, as opposed to vertical scalability which means scaling a database by adding more resources to a single node. One important feature of NoSQL solutions is distribution, which translates in the ability to distribute the database through several nodes, thus scaling the database cluster horizontally. Together with distribution and horizontal scalability comes increased availability and reliability. Distributing a database through several nodes increases availability and reliability, depending on the distribution approach. Several distribution approaches are possible, such as partitioning data and storing every partition only once, which does not guarantee increased data reliability; however, distributing every partition through two or more nodes increases data reliability and availability.

C. **Full-Text Searching Capabilities:** One disadvantage of most NoSQL solutions over relational databases is the inexistence of searching capabilities. Although some NoSQL solutions offer this, most are mostly focused on efficient storage of data and simple retrievals of documents by key. However, to plug this gap several external full-text searching engines were developed, which can then be integrated into these NoSQL solutions. This provides an important advantage of being able to focus entirely on the NoSQL solution without regards to efficient searching capabilities, and being able to externally plug a searching engine, which usually has the focus on efficient searching. This separation means we can replace one searching engine for a more efficient one, without switching the NoSQL storage system, provided that the searching engine provides integration with the storage system. With an external searching engine, it would index context information asynchronously, in a passive way, without compromising the correct functioning of the XMPP PubSub protocol.[5] Through replications, one storage node could be in charge of all store/update functionalities, while a replicated node could be in charge of context information indexing and searching. It should be noted that

existing relational databases such as PostgreSQL already provide full-text searching capabilities natively[6], and do not need external components to do it. It is, however, tightly integrated inside PostgreSQL, so it is not possible to switch it for a more efficient one, or even to separate it completely. Using an external searching engine also provides additional scalability benefits, as most of them allow distributing and may be deployed as a cluster, separately from the NoSQL storage cluster, so different scalability requirements for search should not affect the scalability requirements of the storage system, and both can be thought of being deployed independently.

V. Conclusion

The usage of a NoSQL storage system in an XMPP-based Context Architecture provides important performance advantages, and allows for higher availability and reliability, while also scaling horizontally. The NoSQL ecosystem, unlike relational databases, is headed towards specialization, so different solutions are headed in different directions, leaving the door open for new players to emerge, and making the ecosystem an exciting and ever-evolving field. In this paper, we addressed the impact and improvements a NoSQL solution can have on a Context Management Platform. ACID property is not used in the NoSQL databases because of data consistency so we get to know how SQL lags data consistency. Later, on the basis of the CAP theorem we described different types of NoSQL databases that are Key-Value databases, Document Store Databases, Columnar based databases and Graph databases with the help of an examples Further research is going on in the new technologies that are arising for or after NoSQL that is polygon persistence, etc.

This paper elaborates on non-relational databases, which are subsumed under the umbrella term NoSQL databases. Relational databases are facing a number of challenges due to Web 2.0 applications, distributed architectures and a growing amount of unstructured data. As NoSQL databases address these issues, as they are designed to meet the requirements of Web 2.0. This article analyzes the possibilities of NoSQL databases for spatial planning issues and the spatial domain in general, based on the cadastral document collection. The nature of elements of the cadastral document collection is closely related to unstructured data. Contained documents share similar semantics but differ syntactically which can be explained with the temporal context and the fact that certain items of documents cannot be described with predefined schemata – which is shown based on purchase of land contracts. Additionally, the document collection has a certain dynamics, as a number of update processes take place. Hence, they should be considered as an alternative to relational databases. The ability of Document Stores to query documents addresses this issue. Furthermore, spatial data handling and indexing is implemented in NoSQL databases, which increases their analytical capabilities. Nevertheless, they offer new possibilities for analyzing unstructured spatial data present in distributed environments. Concluding, NoSQL databases have advantages for applications that share characteristics with Web 2.0.Nevertheless, traditional relational databases are designed to support consistent transactions, high security, and complex queries and thus will remain popular. Through availability of a variety of database approaches developers have the freedom to choose the concept that satisfies the application requirements best.

VI. References

[1] Facebook. Facebook Press Room:Statistics. http://www.facebook.com/press/info.php?statistics

[2] SQL and NoSQL Databases: International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 8, August 2012

[3]Nuno Santos, Context Storage Using NoSQL, Instituto de Telecomunicac¸ ˜oes Campus Universit´ario de Santiago

[4] Marc Seeger, Key-Value stores: a practical overview, Computer Science and MediaUltra-Large-Sites SS09,Stuttgart, Germany

[5]Collaborated, Xmpp Protocol, Wikipedia, http://www.wikipedia.org/xmpp

[6] Collaborated, PostgreSQL, Wikipedia http://www.eikipedia.org/postgresql