NoSQL: A Panorama for Scalable Databases in Web

Jagjit Bhatia
P.G. Dept of Computer Science, Hans Raj Mahila Maha Vidyalaya, Jalandhar

Abstract - Various business applications deal with large amount of digital data every day. All the data normally handled by relational model suggested by E.F. Codd’s during 1970. This model has made the benchmarks in database technologies. From 1990 the era of web has started. Enormous web applications were used for different purposes such as business, banking, education and e-commerce etc. Even relational model is also fulfilling the needs of these web applications. But with the passage of time and advancement of web technologies there were immense increase in the user of web applications. The relational model architecture fails to address the problem like scalability, availability of the databases with this advancement. So the companies related to web transaction found in house solution i.e. NoSQL database pronounced as ‘nosequel’. NoSQL solutions are used to design of scalable databases, i.e. the databases having large amount of data and having tendency of databases to increase with respect to time (scalable). In this paper i have discussed various data store technologies used by big web transaction related companies like Facebook, twitter etc.

Keywords - NoSQL, ACID (Atomicity, Consistency, Isolation and Durability), BASE ( Basically Available), BLOB (Binary Large Object), HQL (Hyper Query Language).

I. INTRODUCTION

During 1970 E.F. Codd suggested a relational model to handle data which was used by various business applications and manage their large amount of digital data. This model has made benchmarks in database technologies. This model is well suited for client server programming. The relational database technology was designed for dedicated hardware and storage area network. The database are designed with the concept of single machine having are the responsibility to manage the database and to take care of the consistency of data. These database servers provide the concurrent access to the machine attached to them and there is no concept of data replication because the data replication is s great mingle for consistency of databases. To have a concurrent access on the same data there is no need of any replication of database. The technologies like two phase commit protocol are sufficient to manage concurrency. The relational model architecture fails to address the problem like scalability, availability of the databases. But due to increase of internet usage availability and scalability of databases has become the need and necessity of databases. From 1990 the era of web has started. Enormous web applications were used for different purposes such as business, banking, education and e-commerce etc. Even relational model is also fulfilling the needs of these web applications. But with the passage of time and a advancement of web technologies there were immense increase in the user of web applications. So the companies like Amazon Dynamo, Google BigTable, LinkedIn Voldemort, Twitter FlockDB, Facebook start facing the problem like extraordinary transaction volumes, low transaction execution time to transaction related to large datasets and even reliable services in non reliable environment. These companies found in house solution for above said issues i.e. NoSQL database pronounced as ‘nosequel’. This is non relational database management system derived from relational database. These types of databases are distributed in nature and horizontally scalable. In these databases data is not organized in normalized...
tables [6]. These databases are open source in nature. NoSQL simply not a SQL based Relational database management system.

II. ACID VS BASE

In RDBMS there is study of ACID transactional properties [7] (Atomicity, Consistency, Isolation and Durability). But NoSQL is exemplifying by BASE ellipsis [8]. Basically Available i.e. the data is shared and replicated at different storage servers for available all time if there is failure of one or more servers in environment. Soft state i.e. Data can remain in inconsistent state to reduce complexity and it is the task of application developers to provide the consistency to part of data while using it. Eventually consistent i.e. Data consistency is instantaneous, Like ACID properties, NoSQL does not ensure that the data will remain consistent for future point. With the immense mushrooming of web servers, the data on these servers are scaling with very fast speed. There exist of distributed and scalable databases. In NoSQL databases large data entities are stored as XML documents [9]. XML is a markup language used for web the data or document in XML format is low storage profile as compared to other languages and very much suitable to float on web.

III. CAP PROPERTIES AND SCALABILITY

The CAP theorem [11] states that data system should have three properties (C) consistency: In Distributed environment all the nodes should see same data at a given instant of time. (A) Availability: The data should be available to each node at every instant of time in distributed environment. (P) Partitions tolerance: The system should in working state even in failure of some of its nodes.

These three properties are inversely proportional to each other. There is a tradeoff between consistency and availability. It we try to achieve availability of data by replication it on each one then it will be hard to achieve consistency of database, because one has to update each replica to maintain consistency. Partition tolerance is again related to replica of data more than one node, even failure of some node the availability of data is ensured to reliability of system. Perfect availability and consistency of data is impossible to achieve.

NoSQL solutions are used to design of scalable databases, i.e. the databases having large amount of data and having tendency of databases to increase with respect to time (scalable). To manage such scalable database the concept of distributed databases are used. In distributed databases the data is portioned and stored over the multiple servers which are interconnected with each other rather than a centrally connected storage area network [12]. This is done only to achieve scalability and reliability. The partition of data is replicated to multiple servers to increase the availability of data. If one site fails even then the data will be available on the network from its replica on other site.

Of course the replicas of a data on multiple sites challenge the consistency of databases. This cost had been tolerated by the system to achieve the high end of availability and to manage with the bottlenecks in the system. Here NoSQL is a tool of great use and helpful. NoSQL can generate a strong test bed to find out the punishing situations of the system under critical conditions and even keep on running under unpredictable worst situations arises in the system [13].

It is hard fact that the data stores of NoSQL are often inconsistent. There are various data model of NoSQL. All stores the data in the form of Scalar, Strings, BLOBs and complex reference values. These are termed as tupels, documents, extensible records or an object [14]. A tuple is a row in relational table. Row further consists of attributes. The attributes may be scalar of and are referenced by ordinal position. A document is data stores which allow a wide range of values to
store. Document may be nested or list as scalar values. An extensible record is a hybrid of document and tuples. The defined structure of this record is already stored in schema. But the attributes can be added as per requirement of further extension. Database object is just like an object in programming language without member functions.

IV. DATASTORES IN NoSQL

4.1. Key value stores

By this system data values are stored and managed by indexing on defined attributes. The suitable usage of this technique is adapted in Project Voldemort. It is named after a villain character of famous Novel of Harry Potter. Voldemort is open source written in java by LinkedIn. Voldemort database is not based on relational model. This database does not satisfy any ACID properties of relational model. Neither it is a object base database nor it is document oriented. It is a big scattered, determined, fault tolerant has table. It is horizontally scalable and has high availability with complexity. It allows the replication of data at different servers to increase the availability of data. But it updates the replica asynchronously and does not guarantee the consistency of data. But it provides the updated view by considering majority of replicas. It also supports automatic portioning and replication of data on desired n nodes of the distributed environment. To claim scalability, the nodes can be added or removed from the database cluster. The system acclimatizes these addition and deletion of nodes automatically. About the fault tolerance, it automatically detects the failed node and recovers it. It is based on the concept of key value databases. These databases are combination of two things keys and value. Key is just like a primary key in traditional databases. It cannot be null or replicated. Key is a unique identifier of a particular set of data. For example

| Key | Value |
|-----|-------|
| 1   | From rollno: 7901 to rollno: 8000  
Class: BCA 2nd sem  
Stream: Computer Science |
| 2   | From rollno: 5901 to rollno: 6000  
Class: B.Com 2nd sem  
Stream: Commerce |
| 3   | From rollno: 1901 to rollno: 2000  
Class: B.Sc. (Med) 2nd sem  
Stream: Science |
| 4   | From rollno: 2901 to rollno: 3000  
Class: B.A 2nd sem  
Stream: Arts |

Above Table is consist of two column first column is key and second column represents the ranges of all the Roll numbers of a given class. Here key is arranged in alphabetical order and hash function will be defined to locate the data of a particular roll-on. The major drawback of this model is that the stored data has no schema.

Another application of key value stores is Redis (V.2.4) which is written in C++ language. This application is also based on hash tables assessed by keys. It works in telnet protocol implemented in various client libraries. The client side does the distributed hashing over servers. Servers stores
data in RAM which is stored on disk for backup or system shutdown to add more nodes. It is a open source and BSD (Berkeley Software Distribution) licensed.

4.1.1. Scalaris is another implementation of key stores values. It is similar to Redis. It is written in Enlarg and is open source. It is functionally same as Redis. It does not generate hashing for the nodes. It assigns the ranges of keys to the nodes. By this way the query does not travel to every node and perform the load balancing in a managed way[17].

4.1.2. Amazon’s Dynamo Amazon runs a world-wide e-commerce platform that serves tens of millions customers at peak times using tens of thousands of servers located in many data centers around the world [18]. Amazon platform is restricted with the operational requirements like performance, scalability, efficiency and reliability. Due to involvement of e-commerce the reliability is highly desirable. These properties of the system are dependent on the management of state of system. Amazon platform is fault tolerant, Fail in server or network does not affect the availability of the data and recover the failures without impacting the performance of system as whole. Amazon ensures the reliability and scalability of the system as per needs. The storage technologies like Amazon S3, Dynamo, which are very much popular and simple storage service. Dynamo is most popular technique to achieve availability and scalability.

4.2. Extensible Record Stores: Extensible records are also termed as wide column stores. They can be horizontally and vertically partitioned for distribution across distributed database environment.

4.2.1. BigTable by Google’s is a implemented version of this technique. It basis on the methodology of partitioning the table horizontally or vertically and distribute/replicate it over the multiple nodes. The partitioning is done on the primary key. There are three types of the partitioning which can be done the databases.
   a. Horizontal: Here tables are partitioned by row wise. These partitions are then distributed over the various nodes and may be reunion to get the original data back.
   b. Vertical: Here the table is partitioned on the basis of group of attributes. Each partition must have key attribute and the partition can be joined by using join operation of SQL.
   c. Hybrid: Her both horizontal and vertical partition can be applied on the data.

Another most popular system based on extensible record stores is HBase, which is released in February 2007. This is an Apache project written in java and uses Hadoop file system. In this system there is no client side hashing.

4.2.2. HyperTable is another system written C++ language. It is open source and owned by Zvents. It is similar to BigTable. It uses Distributed file system Hadoop and distributed lock manager. In this system partition is distributed/replicated over the servers by key ranges. It supports the API written in any language. It uses hyper query language (HQL) [16].

4.3. Document Stores: Data is stored in documents which are indexed and fetched by query mechanism. It stores NoSQL databases. Here records are documents which may be text (unstructured) or XML (semi structured). Each document is represented by set of keys document or values documents. All the documents reside in databases and is recognized by a unique field by using hashing. It has no schema and dynamic in nature. It does not provide ACID transactional properties.

4.3.1. CouchDB: CouchDB is based on document stores model. It is Apache project written in Erlang. It is a richer data model. Here document are put in domain. Indexes are created on the field
of a domain. The field may be text, numeric or Boolean or compound. Further indexes are B-tree to
manage the result of queries (views). **CouchDB** does not guarantee the consistency. It implements
Multi Version Concurrency Control mechanism on each document. It also provides the fault
tolerance. During system crash all updates are flushed to disk for commit and also it flushes the disk
for commit after every update of document. Client can interact with **CouchDB** through a special
interface called **RESTful** interface.

4.3.2. **MongoDB**: It is a General programming language open source document store written
in C++. It provides indexes on document domain. It uses lockless mechanism for query processing. It
provides automatic partitioning and distribution of documents on the servers. **MongoDB** does not
ensure the global consistency but support local consistency to primary copy of document in case of
replication. It supports dynamic queries by automatic uses of indices. During the failover it supports
the automatic recovery by master slave replication mechanism. To demand the high performance,
updating of replication is asynchronous. This is a challenge to consistency. Some update may lose
due to system crash.

_Courtesy: http://www.mongodb.com/mongodb_overview.htm_

| DBMS           | MongoDB   |
|----------------|-----------|
| Database       | Database  |
| Table          | Collection|
| Tuple/Row      | Document  |
| column         | Field     |
| Table Join     | Embedded Documents |
| Primary Key    | Primary Key (Default key _id provided by mongodb itself) |
| Database Server and Client | |
| Mysql/Oracle   | mongodb   |
| mysql/sqlplus  | mongo     |

4.3.3. **Terrastore**: Another recently developed document store is Terrastore, based on the
Terracotta distributed Java VM clustering. In this system client request is based upon the HTTP
operations to retrieve or update the data. It uses java and Python Application programming interface.
It automatically partition and distribute the data over the network due to addition and removal of any
node from the environment. It is schema less and does not follow ACID properties. It provides
high range of scalability and flexibility to a database without sacrificing a consistency of the
databases.

V. Conclusion

We consider three techniques to manage large database with NoSQL. a). Key Value Stores b).
Extensible Record Stores c). Document Stores. All the three have it own properties, advantages
and limitations too. Every data stores have its own area of application depending upon its properties
and need of the requirement. Key-value stores are suitable solution for application having one kind
of object. It is suitable for the web application generating the page by using several SQL queries
during login. We can manage this application by key value stores to avoid from many RDBMS
queries. If we have to deal with application having different kind of objects like Student databases, where data retrieval based on multiple fields like rollno, According to Class , According to Stream, According to Subject, According to Birth date. The document stores do not guarantee consistency. If consistency of data at various site is not an issue for a system concerned then this stores work well. It provides the eventual consistency with limited atomicity to the databases.

The usage of implementation for extensible record store is also for the application using multiple kinds of objects. But it aimed at higher performance. It provides higher level of consistency with a cost of complexity. Here is portioned horizontally and vertically for the site or server in network according to need and demand of the data at particular site. The cost emerges to manage the local schema and global schema overall the all sites. Many e-commerce site which has reliability as basic need uses these data stores.

REFERENCES

[1]. Amazon Dynamo: http://www.allthingsdistributed.com/2007/10/amazons-dynamo.html.
[2]. Google BigTable: http://labs.google.com/papers/bigtable.html.
[3]. LinkedIn Voldemort: http://project-voldemort.com/.
[4]. Twitter FlockDB: http://engineering.twitter.com/2010/05/introducing-flockdb.html.
[5]. Facebook Cassandra: http://cassandra.apache.org/.
[6]. Yahoo! PNUTS: http://www.brianfrankcooper.net/pubs/pnuts.pdf.
[7]. SilvanWeber, “NoSQL Databases”: http://www.christof-strauch.de/nosqldbds.pdf
[8]. Thompson J., “Oracle NoSQL Database”, An Oracle White Paper, September, 2011.
[9]. Andrew J. Brust, “NoSQL and the Windows Azure platform”, Blue Badge Insights, Inc., April 25, 2011
[10] DAMA - Philadelphia / Delaware Valley, the “Role of Data Architecture in NOSQL”, Wednesday January 11th, 2012, http://www.damaphila.org/HaugheyNOSQL.pdf
[11] E. Brewer, “Towards Robust Distributed Systems”, Proc. 19th Ann. ACM Symp. Principles of Distributed Computing. (PODC 00), pp. 7-10, ACM, 2000.
[12] Jeremy Zawodny, “Lessons Learned from Migrating 2+ Billion Documents at Craigslist”: http://www.10gen.com/presentation/mongodb2011/craigslist
[13] M. Brantner et al., “Building a Database on S3”, Proc. ACM SIGMOD Int’l Conf. Management of Data (SIGMOD 08), pp. 251-264, ACM, 2008.
[14] http://www.project-voldemort.com/voldemort/
[15] Abhinay et al., “Growth of New Databases & Analysis of NOSQL datastores”. International Journal of Advanced Research in Computer Science and Software Engineering 3(6), pp. 1306-1319, June - 2013.
[16] Rick Cattell, “Scalable SQL and NoSQL Data Stores”, Originally published in 2010, last revised December 2011.
[17] Vatika Sharma et al.,” SQL and NoSQL Databases”, International Journal of Advanced Research in Computer Science and Software Engineering 2 (8), pp. 20-27, August- 2012.
[18] B. DeCandia et al, “Dynamo: Amazon’s Highly Available Key-Value Store”, Proceedings 21st ACM SIGOPS Symposium on Operating Systems Principles, 2007.
[19] M. Stonebraker and R. Cattell, “Ten Rules for Scalable Performance in Simple Operation Datastores”, Communications of the ACM, June 2011.