ISSN No. 0976-5697

Volume 5, No. 5, May-June 2014

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A Comparative Study of NoSQL Databases

Abhishek Prasad[1], Bhavesh N. Gohil[2]
[1,2]S.V.National Institute of Technology,
Ichchhanath, Surat,India

Abstract:In today's scenario, web applications are facing new challenges while serving millions of users. It can be easily understood and appreciated that while accessing these web applications from all over the world, users expect the services to be always available with high performance and reliability. The rate of growth of successful web services is much faster as compared to increase in the performance of computer hardware. Therefore, after sometime a web application which attempts to provide large number of web services and support large number of users needs the ability to scale. This study explores information on various NoSQL (Not only Structured Query Language) databases and attempts to make a comparison between them based on different criteria. The NoSQL databases were created to offer high performance and high availability especially for large scale and high concurrency applications at the expense of losing the ACID (Atomic, Consistent, Isolated, Durable) properties of the traditional databases in exchange with a weaker BASE (Basic Availability, Soft state, Eventual consistency) features.

Keywords: Big Data;NoSQL; Scalability; Sharding; Replication; Comparison

I. INTRODUCTION TO NOSQL DATABASE

A. *Understanding of NoSQL database:*

a. What isNoSQLdatabase?:Since last 15 years or so, interactive applications have changed dramatically which has led to the change in the data management requirement of those applications. Nowadays, three interrelated megatrends: Big Data, Big Users and Cloud Computing are driving the adoption of NoSQL technology. Therefore, NoSQL is being increasingly considered as a feasible alternative to relational databases, especiallyasthey can simplify operating on large scale providing betterresults on clusters of standard and commodity servers.Further, it is opined that a schema-less data model is better for the capturing a variety of data processed today.

b. Why NoSQL hasLargeUsers?:Today, with the rise in global Internet use, there has been considerable increase in the number of users and the time theyspend online. Smartphones and tablets have started gaining popularity, as internet is easily accessible anywhere through these easy-to-handle devices.

It is important to support large numbers of concurrent users.At the same time it is also significant to dynamically support rapidly growing (or shrinking) number of concurrent users.

To support large number of users in addition to the dynamic nature of application usage patterns, there is a need for more easily scalable database technology. With relational technologies, it is difficult to get dynamic scalability while maintaining the performance users demand for their application.

c. NoSQL can handle vast data:With the advent of ever growing technology, it is easier to capture data by accessing third parties sources such as Facebook. With a single click one can capture personal user information, geo location data, social graphs, machine logging data, and sensor-generated data. Data is very huge and ever-expanding.Use of the data can rapidly change the nature of communication, shopping, advertising, entertainment, and relationship management. It may also happen that applications which don't leverage it quickly and timely will lag behind in no time.

d. NoSQLbetter suited for cloud computing:Presently, most new applications use three-tier Internet architecture. These applications run in a public or private cloud, and support large number of users.In three-tier architecture, applications are accessed through a web browser or mobile application connected to the Internet. In the cloud, a load balancer directs the incoming traffic to a scale-out tier of web or application servers that processes the logic of the application.

Traditionallyrelational databases were the popular choice for database tier. These relational databases arecreating a lot of problem. But these are still used, because they havepeculiar features of being centralized, share-everything technology that scales up rather than scalingout. They are not fit for applications that require easy and dynamic scalability. NoSQL databases are more suited in such applications since they provide distributed, scale-out technologies.Therefore,NoSQLis a better option for highly distributed nature of the three-tier Internet architecture.

B. *Common Characteristics of NoSQL:*

NoSQL databases share a common set of characteristics as mentioned below:

a. No Schema required:Data can be inserted into NoSQL database without first defining a rigid database schema. Also, the format of the data being inserted can be changed at any time, without disrupting the application. This characteristic provides immense application flexibility.

b. Auto-sharding:A NoSQL database automatically spreads data across servers, without requiring applications to participate. Servers can be added or removed from the data layer without application downtime, with data automatically spread across the servers. Most NoSQL databases also support data replication, storing multiple copies of samedata across the cluster and even across data centers to ensure high-availability and support disaster recovery.

c. ***Distributed query support:***"Sharding" relational databases can reduce the ability to perform complex data queries operation. NoSQL database systems retain their full query expressive power even when data is distributed across huge number of servers.

d. ***Integrated caching:***To increase thedata throughput and reduce latency, advancedNoSQL database technologies transparently cache data in system memory.

e. ***Reliability (Fault Tolerance):***If some machines within the system crash, the rest of the machines remain unaffected and work continues without any stoppage.

f. ***Scalability:***In distributed computing, more machines can be easily added to the system according to the requirements of the user and the application.

g. ***Sharing of resource:***Similar to data or resources shared in distributed system, other resources can be also shared like expensive printers.

h. ***Speed:*** A distributed computing system can provide more computing power. It can provide parallel computations.

i. ***Performance:*** The collection of processors in the system can provide higher performance than a centralized computer. This also gives better price/performance ratio.

C. High Level Comparison with SQL Databases:

Table 1: Comparison between SQL and NoSQL database [4]

	SQL Databases	*NoSQL Databases*
Types	One type with minor variations between them.	Many different types of databases including key-value, wide-column, document databases, and graph databases.
Data Storage Model	Records are stored as rows and columns where each column stores specific data about that record. Separate data types are stored in separate tables which are joined together when complex queries are executed.	Varies based on type of database. For example, key-value, Document databases store all relevant data together in single "document" in JSON, XML, or another format, which can nest values hierarchically.
Scaling	Vertical scaling: a single server is made increasingly powerful to meet the increased demand. It is possible to spread SQL databases over many servers, but requires significant additional engineering.	Horizontal scaling: simply add more commodity servers or cloud instances to increase capacity. The database automatically spreads data across servers as necessary
Supports Transactions	Yes, updates can be configured to complete entirely or not at all.	In certain circumstances and at certain levels (e.g., document level vs. database level)
Data Manipulation	Structured query language using Select, Insert, and Update statements,	Through object-oriented APIs
Consistency	Can be configured for strong consistency.	Depends on product. Some provide strong consistency (e.g., MongoDB) whereas others offer eventual consistency (e.g., Cassandra)

II. REVIEW OF NOSQL DATABASES

A. SimpleDB:

Amazon provides a web services in the form of Simple DB and distributed data base. SimpleDB does not need any rigid structure and is easy to use. Since, SimpleDB a structure free , it decides of its own about the requirement of indexes and accordingly provides a simple SQL like query interface. [2]

In SimpleDB , to ensure the safety of data and also to increase the performance, all data are replicated onto different machines in different datacenters. Because of this special feature, there is no possibility of automatic sharding .It leads to safeguarding of data from scaling. Scaling is possible only in default where in the application layer has performed the data partitioning by itself. [2]

Some of the salient features of Simple DB which encourages /discourages its use are as mentioned below:

a. ***Consistency:*** SimpleDB provides eventual consistency but does not provide MVCC due to which conflicts cannot be detected on the client side.

b. ***Limitations:***Like any system, SimpleDB also has some limitations in term of capacity of domains, inquiry time, and quantity of domains it can handle. These limits are not rigid and are likely to change as long as system is in use. Some of restrictions are as tabulated below:

Table 2: Parameter and limitations of Simple DB [2]

Parameter	*Limitation*
Domain Size	10 GB per domain / 1 billion attributes per domain
Domains per Account	100
Attribute value length	1024 bytes
Maximum items in Select response	2500
Maximum query execution time	5s
Maximum response size for one Select statement	1MB

B. Dynamo:

Amazon uses Dynamo for the purpose of its own internal application .Dynamo is a distributed key-value storage system. While performing its function Dynamo provides an inquiry API, thus allowing customers to recover a value for a unique key and to put key-value pairs into the storage with values less than one megabyte. [2]

In Dynamo, the work load is dispersed based on the capacity of the nodes. Irrespective of position, each load performs same task. To safe guard the complete data centre from any catastrophe,every key-value pair is duplicated with a geological distribution over several data-centers throughout the world. [2]

Dynamo can provide availability, consistency, cost-effectiveness and performance. Dynamo uses optimistic replication with multi-version concurrency control (MVCC) to achieve a type of ultimate stability. It has properties of both databases and distributed hash tables (DHTs).

a. ***Sharding:***Dynamo uses various types of consistent hashing to partition data. Virtual nodes are created by sub dividing each node, and finally these virtual nodes are linked with a random position in the key space. One can define key space as a group of numbers into which every key can be mapped with a hash function.

A node is accountable for all keys positioned between itself and its predecessor. [2]

b. ***Replication:***The working pattern of Dynamo is that it dictates number of replicas a key should have with the number. Concerned node for k replicates k to its N-1th successors. Then MVCC is used for letting the asynchronous synchronization between the replicas. Every time, a new revision is created. Vector clocks are used for implementing the revision control of Dynamo. [2]

c. ***Consistency:***Dynamo uses a configurable quorum like approach. The client can specify the number of replicas R, a key must be read and number of replicas W must be produced during a write. With $R + W > N$, would be equivalent to a quorum system with strong consistency. With $R+W < N$, would be in favor of latency and availability over consistency. [2]

C. MongoDB:

MongoDB is a structure free, cross-platform document oriented database developed by 10gen. It has an open source neighborhood [5]. The name MongoDB has originated from "**humongous**". The database is meant to be scalable and fast. Mongodb is written in C++. Large binary files like images and videos can be stored and distributed by using MongoDB . Each document is assigned an ID field, which acts as a primary key. MongoDB stores documents as BSON (Binary JSON) objects, which are binary encoded JSON like objects[6]. MongoDB can also be effectively used for indexing over embedded objects and arrays.

a. ***Replication:***Repica set can be defined as a group of Mongod occurrences that host the same data set. One mongod, the primary receives all write operations. All other occurrences, secondaries apply operations from the primary so that they have the same data set. The primary accepts all write operations from customers. Replica set can have only one primary. Replica sets provide rigid stability. The primary logs all changes to its data sets in its oplog to support replication.

b. ***Sharding:***MongoDButilisesautosharding, which takes care of irritants normally created during manual sharding. The cluster handles division of data and rebalances automatically.

A MongoDB cluster has three components; Shard nodes, configuration servers and routing services known as mongos. Mongos are independent and therefore can be employed to function in parallel

c. ***Consistency:***To facilitate reading an old version of an executed update operation, MongoDB provides concluding consistency to the respective process. [2]. MongoDB has no version concurrency control and no transaction management.

d. ***Failure Handling:***MongoDB is designed in such a way that texts are written to the journal within 100 milliseconds [5]. The written text is “resilient” meaning that the data will still be recoverable even after a pull-plug-from-wall and a hard restart. Inspite of the fact that the journal commit is nearly instantaneous, MongoDB takes time to write the data files. Normally, by default MongoDB may take a minute to write data to the data files [5]. The slow speed of MangoDB does not affect stability because the journal has adequate material to ensure recovery from sudden loss[5]. Depending upon requirement, it is possible to alter the time gap for writing to the data files.

D. CouchDB:

CouchDB is a non-structured document dependent database with a favourable replication mechanism. The project is written in Erlang and is part of the Apache Foundation [2]. CouchDB is a NoSQL database which uses JSON to store data, JavaScript its query language using MapReduce, and HTTP for an API. It has a unique feature of multi-master replication. In CouchDB a unique identifier is used for its document. In CouchDB, updating process is carried out on whole documents. Revision id is used in concurrency control of the document. One other interesting feature in CouchDB is the option to specify a validation functions in JavaScript. Couch DB has a unique feature which enables it to specify validation functions in JavaScript.

a. ***Replication:***In replication process, CouchDB compares the database of source and the destination to determine the difference of documents between the source and the destination. Destination documents of same revision which already exist are not transferred.

A replication task will finish once it has reached the end of the changes fed in the system. If its continuous property is set to correct figure, it will wait for new changes to appear or the task is abandoned. Replication tasks also generate check point documents on the destination.

Both nodes can be used for writing and reading in a master-master setup. Such a setup will not have effect on partition. It is possible to synchronize CouchDB with more than one other database [2].

The shortcoming of optimistic replication is its inability to handle conflicts between different revisions of the same object [2].

CouchDB uses the benefit of hash histories to identify duplicate versions of the same document; this means that an update of a document only changes the revisionid [2].

b. ***Sharding:***CouchDB does not have its own built in sharding mechanism yet, but at present there are two projects which provide sharding support for CouchDB.

The first project is The Lounge [2], a tool set that contains proxies, which can dispense the data to different CouchDB instances. The Lounge is developed and used by Meebo.com.

The Cloudant[2] is another project that provides data partitioning for CouchDB .The Cloudant provides a hosting service with automatic sharding functionalities for CouchDB databases.

c. ***Consistency:*** CouchDB database consistency is dependent on how CouchDB is used. CouchDB provides eventual consistency if its built in replication feature is used in a master-master configuration. But the replication can also be used in a master-slave setup, which would provide strong consistency. [2]

d. ***Failure Handling:***CouchDB has no method to deal with failures of nodes. Currently it only provides the replication mechanism. CouchDB uses an append only B+Tree as file structure to be strong against power disruptions and to minimize the number of seek operations of the hard drive while writing into the database. The copy-on-write characteristic of CouchDB makes sure that the files on the disk are always in a constant state, even if a power failure occurs. Because of this, CouchDB does not need a transaction log file and can always be disrupted without the risk of damaging the database files. [2]

E. BigTable:

BigTable is one of Google's techniques to deal with its vast amount of data. It is built on top of Google's distributed file system GFS[2] and is used by Google for several applications with different demands on the latency of the database.

BigTable is tolerant to hardware failures as it is designed to be scalable, distributable and therefore Google's own implementation is not available to everyone, besides the usage inside Google's App Engine.

a. ***Sharding:***The tables within a BigTable databases are divided by their row-key into several key-ranges, which are called tablets .Every tablet is allocated to only one tablet server at any instant. A master server stores the meta information regarding the existing assignment of each tablet and allocates the currently unassigned tablets to currently available tablet servers.[2]

b. ***Replication:***BigTable does not directly replicate the database because one tablet can only be assigned to one tablet server at a time, but it uses Google's distributed file system GFS[2] for the storage of the tablets and log files, which handles the replication on the file level.

c. ***Consistency:***BigTable can provide robust consistency because its each tablet is assigned only to a single tablet server. This limits the availability due to the eventual limited accessibility of tablets during the recovery from the loss of a tablet server. BigTable permits creation of atomic operations on one row and also offer the ability to make transaction inside a row. It is not possible in BigTable to make transactions over multiple rows.[2]

d. ***Failure Handling:***In BigTable, log files are used for all write operations which are akin to the transaction log files of relational databases. Each tablet server writes its own log file to GFS. Therefore, during the recovery phase each tablet server of a previously lost tablet server has to read its log file and search it for write operations on the tablets.

There could be another failure scenario where in the master fails. In such situation, tablet servers choose a new master server using the Chubby service and restore the meta-data tablets. This is carried out by using the root tablet file in the Chubby service. Five servers are there to ensure the availability of the Chubby service, and at least three of them are required to get an exclusive lock on a Chubby file.[2]

F. Cassandra:

Cassandra is basically a hybrid between a key-value and a tabular database.

A column family is similar to a table in an RDBMS. Unlike a table in an RDBMS, different rows in the same column family do not have to share the same set of columns. Further a column may be added to one or multiple rows at any time.[7]

In Cassandra, each key corresponds to a value which is an entity. Each key has values as columns, and columns are clustered together into sets called column families. Then, these column families could be regarded as tables. A table in Cassandra is a distributed multi-dimensional map indexed by a key.[7]

Furthermore, applications can identify the sort order of columns within a Super Column or Simple Column family.

a. ***Replication:***Replication is used in Cassandra to achieve high availability and durability. Cassandra offers the client with various alternatives as to how data needs to be replicated. Application decides replicas which are chosen based on the Replication policy. Cassandra system elects a leader amongst its nodes using a system called Zookeeper[8]. A node is responsible for certain ranges of metadata which is cached locally at each node and in a flawless manner inside Zookeeper. In this manner if a node which has crashed and retrieved subsequently is aware of ranges it was responsible for.

By relaxing the quorum requirements, Cassandra offers durability assurance in case of node failures and network partitions. Malfunctioning of data center may happen due to power disruptions, cooling failures, network failures, and natural disasters. Preference list of a key are actually constructed in such a manner that the storage nodes are extended across multiple datacenters.

b. ***Consistency:***Cassandra can be everything from a strict consistent data store to a highly eventual consistent system. In the case of a strict consistent data store every read requests always gets the latest data where as in the case of a highly eventual consistent system old data is sometimes provided for a huge gain in availability and performance. Any configuration where $R+W > N$ is considered strong consistent. Here, R is the amount of replicas which are inquired for a read operation, W is for writes and N is the replication count.[1]

c. ***Sharding:*** Cassandra has an exclusive design feature where in it has the ability to scale incrementally. This requires the ability to dynamically partition the data over the set of nodes (i.e., storage hosts) in the cluster. Cassandra uses consistent hashing to partition data across the cluster. In such a case, it uses an order preserving hash function for this purpose.[8]

The basic algorithm is oblivious to the heterogeneity in the performance of nodes. This issue can be addressed in two ways .In first method, nodes are assigned to multiple positions in the circle (like in Dynamo).In second method, load information on the ring are required to be analyzed and lightly loaded nodes are moved on the ring to ease heavily loaded nodes. Cassandra prefers second method as it makes the design and implementation very easy and helps to make clear cut choices about load balancing.[8]

G. Riak:

Riak was written byAkamai5 engineers and is published both as a feature to reduce open source version and a full featured commercial version. The company "Bashoo" markets the commercial version along with support, maintains and writes almost all of the source code. In majority of cases, Riak is written in Erlang while some parts are also written in JavaScript. Clients are at liberty to correspond with Riak using a REST-API or via Google's Protocol Buffers. [1]

a. ***Sharding:***Riak scales linearly with the addition of nodes. This results in improvement in terms of reliability, performance and throughput with larger clusters. It is recommended to deploy five nodes or greater to provide a base for high performance and growth as the cluster size increases. When a new node is added to Riak cluster, the data is rebalanced automatically without any downtime. A user does not have to deal with underlying complexity of data location as any node can accept or route requests. [9]

b. ***Replication:***Riak Enterprise provides multi datacenter replication, monitoring and 24×7 support. Raik enterprise is an extension to riak to replicate data across availability zones. Users can use multi datacenter replication to serve global traffic, run secondary analytics clusters or meet disaster recovery and regulatory requirements and maintain active backups. Multi datacenter replication can be used in more than one site. Riak Enterprise provides two ways for multi datacenter replication: 1) full sync; 2) real-time sync. In case of full sync, data is replicated at scheduled intervals between two clusters. Default is six hours. In case of real-time sync, Updating of primary data center triggers replication to the secondary data center. After writing an object to the primary cluster, writes are sent to the secondary cluster via post commit hook. All multi datacenter replication take place over TCP connection and supports SSL.[9]

c. ***Consistency:***Raik supports eventual consistency.

d. ***Failure Handling:***Data is not lost when access to many nodes are lost due to network partition or hardware failure loses. Riak replicates key/value stores across a cluster of nodes with a default n_val of three. Data can be written to a neighboring node beyond the initial three, and read-back due to its “masterless” peer-to-peer architecture in case the node outages due to network partition or hardware failures, [10].

H. Apache Hbase:

The first release of Hbase was made in 2007. It was the part of Apache Hadoop project. It became an Apache top level project in 2010 and was released independently from hadoop.

Most of the concepts and services used in Big Table is similar to HBase, such as GFS, data model and Chubby. All HBase modules are written in Java language. Testing of Hbase modules are performed on Linux-Systems and not on Windows.

a. ***Replication:***Asynchronous replication is used in Hbase where clusters can be geographically distant from each other, the links between the clusters can be offline for some time and rows inserted on the master cluster won’t be immediately available on the slave clusters. This is also called as eventual consistency.

HBase replication on each region server is based on HLogs. These HLogs are stored in HDFS to replicate data to any slave cluster.

The size of clusters participating in replication can be asymmetric and the master cluster puts its best to balance the stream of replication on the slave clusters by using a concept of randomization.

b. ***Scaling Mechanisms:***In HBase concept of replication count is not there. Each row of a table is stored on exactly one Region Server. The setup can be scaled by adding more number of nodes to the cluster. This helps in reducing load on the servers and provides additional space. Adding more nodes speeds up both read and write operations on the cluster. The performance of the HDFS is dependent on the performance and availability of HBase. [1]

c. ***Consistency:***HBase can guarantee strong consistency using its single Master setup. As all read and write operations are immediately visible to all the clients connected to master node no outdated data is returned. Both the Region Servers and the Master have single points of failures. If one fails, some part or the complete data is unavailable for a short period of time until that node is replaced by the new node. [1]

d. ***Failure Handling:***There are two different types of errors that can occur on aHBase system.

a) ***Master Fails:*** A new Master can be elected. No downtime is required using Automatic failover.

b) ***Region Server Fails:*** A new node is assigned to that region by the Master. This new machine will read the data image and the commit logs from HDFS and can then take over.[1]

I. Redis:

Redis made its first release as open source project in 2008 written in C language. The main idea was to provide a key value store. Redis provides a simple interface where different data types can be assigned to a key. Later on, the data can be requested again using the key. For improving the performance Redis tries to maintain all the data in the systems main memory. If enough space is not available then some parts can be swapped to disk using a virtual memory system.

a. ***Scaling Techniques:***Redis is designed as a single node system with some extensions to make the system scalable.

b. ***Replication:***Redis supports unidirectional replication. The slave replicates all the data from the master node asynchronously and the slave is outdated for some time. This makes the slave a read-only node. Multiple slaves can be set up for each master node. A multi-level replication can also be set up by replicating the nodes again. In Replication all data is copied and therefore no selective replication on a subset of the data.[1]

c. ***Sharding:*** Sharding can be used to reduce load from a Redis setup or to store more data that cannot be handled by a single node. Sharded cluster can be setup by dividing the key space into same number of parts as there are nodes on the cluster. To ensure availability, additional nodes can be deployed and set up as replicas of the primary nodes.[1]

III. NOSQL DATABASE COMPARISON

All examined NoSQL Databases share the same restrictions, which are derived from their distributed architecture. All of them are victims of the CAP theorem and therefore can either only provide eventual consistency or sacrifice some amount of availability.

The examined NoSQL databases have small but very significant differences.

A. *Generalcomparison.*

Table 3: General Comparisons of the system which were examined

8	*Programming language*	*License*	*First Release*	*Platform*
SimpleDB	-	Propriety	2007	Linux, Mac OS X, Windows
Dynamo	Java	Part of AWS	-	Linux
Apache Cassandra	Java	Apache License 2	July 2008	Linux, Mac OS X, Windows
Basho Riak	JavaScript	Apache License 2	2009	Linux, Mac OS X, Unix
Apache HBase	Java	Apache License 2	2007	Linux, Mac OS X, Windows
Redis	C	New BSD	March 2009	Linux, Mac OS X, Windows
CouchDB	JavaScript	Apache License 2	2005	Linux, Mac OS X, Windows
MongoDB	C++	GNU AGPL v3.0	2008	Linux, Mac OS X, Windows
Google Big Table	C, C++	Propriety	2005	Linux, Mac OS X, Windows

B. *Features based comparison:*

Table 4: Features based comparison of the systems

	Replication	*Sharding*	*Consistency*	*PartialTolerance*
SimpleDB	Yes	Yes(at application layer)	No	No
Dynamo	Yes	Yes	Yes	No
Cassandra	Yes	Yes	No	Yes
Riak	Yes	Yes	No	Yes
HBase	By underlying DFS	Yes	Yes	Yes
MongoDB	Yes	Yes	Yes	Yes
Redis	Yes (unidirectional)	Using external tools	Yes	Yes
CouchDB	Yes (bidirectional)	Using external tools	No	Yes
BigTable	Yes	Yes	Yes	Yes

C. *Summary:*

Table 5: Based On Major Users, Storage Type, Best Use and Key Points

	Major Users	*Storage Type*	*Best Use*	*Key Points*
SimpleDB	-	Document	Solutions for simple databases	Provides Amazon support and documentation. Limits on domain size and query time.
Dynamo	-	Key-Value	large to big db sol.	-
Cassandra	Facebook, Twitter, Digg	Column	write often, read less	A cross between BigTable and Dynamo. High availability.
Riak	Mozilla, Comcast, AOL	Key-Value	high availability	Riak is a truly fault-tolerant system which has no single point of failure
HBase	Facebook	Column	random read write to large database	Capable of storing huge quantities of data – Modeled after Big Data
MongoDB	Craigslist, Foursquare, Shutterfly, Intuit	Document	Statistical Data rarely read, frequently written and Dynamic queries	Retains some properties of SQL such as index and query
Redis	StackOverflow	Key-Value	Statistical Data rarely read, rapidly changing data, frequently written	Very Fast
CouchDB	LotsOfWords .com	Document	Changing data with pre-defined queries	DB consistency, easy to use
Google BigTable	Google	Column	Scale across thousands of machines	Currently not used or distributed outside of Google, but can access through Google App Engine

IV. CONCLUSION

Different NoSQL databases were compared based on different aspects. Different properties such as Replication, Sharding, Consistency and Failure handling were discussed for different NoSQL databases. The best NoSQL database for each of the features was selected and shown below.

Table 6: Best NOSQL databases for different features

Aspects	*Best Database*
High availability	Riak, Cassandra, Google Big Table, Couch DB
Partition Tolerance	MongoDB, Cassandra, Google Big table, CouchDB, Riak, Hbase
High Scalability	Google Big table
Consistency	MongoDB, Google Big Table, Redis, Hbase
Auto-Sharding	MongoDB
Write Frequently, Read Less	MongoDB, Redis, Cassandra
Fault Tolerant (No Single Point Of Failure)	Riak
Concurrency Control (MVCC)	Riak, Dynamo, CouchDB, Cassandra, Google Big Table
Concurrency Control (Locks)	MongoDB, Redis, Google Big Table

V. ACKNOWLEDGMENTS

It has been a great experience while undertaking research on **“A Comparative Study of NoSQL Databases”** because it gave me an opportunity for immense learning, value addition and also enriches my knowledge.

I take the opportunity to express my profound gratitude and deep regards to Mr. Bhavesh N. Gohil, my mentor for his keen and sustained interest. He kept guiding, monitoring and continuously encouraging for qualitative completion of this assignment.

VI. REFERENCES

[1]. Dominik Bruhn, “Comparison of Distribution Technologies in Different NoSQL Database Systems”, Institute of Applied Informatics and Formal Description Methods (AIFB), Karlsruhe Institute of Technology (KIT).

[2]. Kai Orend, “Analysis and Classification of NoSQL Databases and Evaluation of their Ability to Replace an Object-relational Persistence Layer”, Master Thesis, Technical University of Munich, Munich,2010

[3]. http://nosql.findthebest.com/ [Online; accessed 10-April-2014]

[4]. Mongodb, http://www.mongodb.com/nosql-explained [Online; accessed 14-April-2014]

[5]. 10gen.com: Home - MongoDB. http://mongodb.org/ [Online; accessed 6-April-2014]

[6]. 10gen.com: BSON - MongoDB. http://www.mongodb.org/display/DOCS/BSON, 2009, [Online; accessed 10-April-2014]

[7]. Wikipedia:`en.wikipedia.org/wiki/Apache_Cassandra[Online ; accessed 20-March-2014]

[8]. http://www.datastax.com/documentation/articles/cassandra/cassandrathenandnow.html[Online; accessed 25-March-2014]

[9]. Brian Holcomb, NoSQL Database in the Cloud: Riak on AWS, June 2013

[10]. Wikipedia: en.wikipedia.org/wiki/Riak[Online; accessed 5-April-2014]