www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 1 January 2016, Page No. 15631-15637

Comparison and Analysis of Document Stored Databases

Shriram Sharma, Atul Chaudhary (Asst. Prof.)

Department of Computer Science & Engineering
Govt. Engineering College, Ajmer, India
grs.sharma@hotmail.com

Department of Information Technology
Govt. Engineering College, Ajmer, India
atul.chaudhary82@gmail.com

ABSTRACT
NoSQL (Not Only SQL) technology includes broad variety of different databases technologies that were developed in response to storage of large volume of user data, handling high access frequency, performance of system and processing of the data. Relational databases were not designed to deal with scalable modern real time applications and agility challenges faced by these applications. RDBMS are used in many applications for long time, the data is stored in tabular form and it is stored in meaningful way, but now there is need to store and manage large amount of data which cannot be handled by traditional relational databases. NoSQL technology is used to overcome this feature of the traditional databases by providing efficient way of storing and managing various types of data with huge amount of dataset. In this report performance analysis is done on document oriented databases: MongoDB, CouchDB and Cassandra. Document oriented database is category of the NoSQL databases where the data is stored in JSON like files. This makes the database capable of storing huge amount of data anywhere in the disk.

Index Terms— NoSql Databases, Mongodb, CouchDB, Cassandra, Big Data.

1. INTRODUCTION
The NoSQL databases provide a medium for storage and retrieval of large data that is modeled in non-tabular form instead of tabular relations used in relational databases. The data structures used in NoSQL databases are of various types rather than tabular form used by relational database management systems, eg. document, graph, key-value etc. These databases are schema less, which makes them stupendously great in performance.
NoSQL databases are better than relational databases because these are more scalable and provide extraordinary performance [1].There are mainly three classifications of data structured, semi-structured and unstructured data. Relational databases can only handle structured data; it cannot handle other two types of data. Relational databases need information of the data before actually storing it in form of schemas, which fits poorly with agile development approaches, because there will be a need to change the schema each time whenever there is need of new features which slowdowns the process if the database is large.
This report makes an attempt to analyze the execution time of queries (of extracting or inserting data) into these document-oriented databases MongoDB, Cassandra and CouchDB.

1.1 Document Database: Document databases are one of the mostly used and popular NoSQL systems, where each record is thought of as a "document". Documents are used to store group of data that convert some sort of user-readable information to the standard formats i.e. JSON, XML, BSON etc. These databases are a subclass of key-value databases. [2]
To maintain locality of data document is the best alternative because these are independent units which make performance better because the related data is read contiguously off disk. It also makes the distribution of data across multiple servers easy. In these system there is no need to translate objects of applications to SQL objects. The developer can easily use the object model directly into a document.
The storage of unstructured data is easy because document contains only those keys and values which application logic requires. It also provides great flexibility by not knowing information schema in advance. [2]
In this report we are going to compare document databases MongoDB, Cassandra and CouchDB on various parameters.

1.1.1 Mongodb: MongoDB is cross platform NoSQL document database. MongoDB is written in C, C++,JavaScript. It was first developed by the software company 10gen (now MongoDB Inc) and shifted to open source community in 2009.MongoDB is widely used by many companies such as Forbes, Bosch, MetLife etc.
MongoDB data model which describes how the semi-structured data is stored in documents as various fields. Collection is a group of documents whereas database is a group of collections. This simplifies the understanding of the databases. [4]
MongoDB stores the data in documents similar to JSON. It is very flexible document data model which contains one or more fields. These fields can include arrays, binary data, and sub-documents. The selection of fields in a application can vary according to the requirement. This feature allows developers to change the data model frequently as their application requires. Documents can be accessed through rich drivers available in almost all popular programming languages

such as Java, PHP etc. MongoDB removes the need of separate ORM layer which means developers need not to handle mapping of objects from database to application which makes them more productive.

MongoDB allows auto-sharding for horizontal scaling of database. To provide high availability across data centres it provides replication. Replication means storing copy of data (secondary set) in servers. If at any time primary set of data goes down, the secondary set automatically takes over as primary set of data. MongoDB also provides in-memory mechanism to speed up the operation by extensive use of RAM (Random Access Memory).The working of MongoDB further can be explained by the following fig.

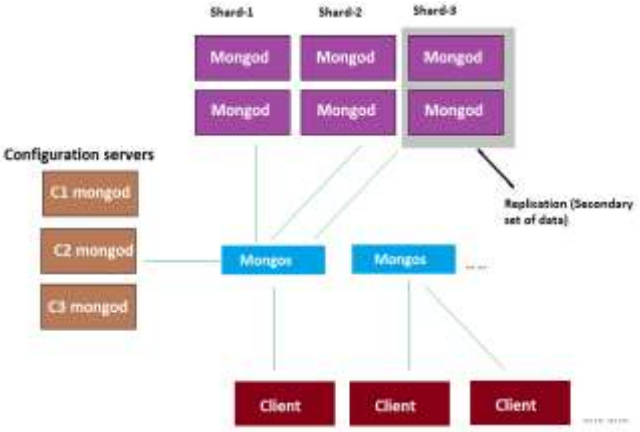

Fig.1 Working of MongoDB

The fig shown above shows the typical auto-sharding used by MongoDB to ensure high availability of data. It also shows replication process which is used for storing same data copy in one or more servers.

1.1.2 CouchDB: CouchDB is document oriented database that completely embraces the web. It was first released in 2005 and later in 2008 became Apache project. It is written in Erlang programming language. CouchDB stores user's data with JSON documents. It uses JavaScript for MapReduce indexes which behave as a query language to the database. It provides HTTP for an API (Application Programming Interface). CouchDB can be queried with web browser via HTTP. An application may access user's mobile database or database on server as per the requirement. [5]

CouchDB removes the need for a server side middle layer which allows a client application to talk directly to the CouchDB, results in reduced development time. Demands can be handled by adding more replication nodes. Replication of database can be done at the client side which means users can perform the operation offline.

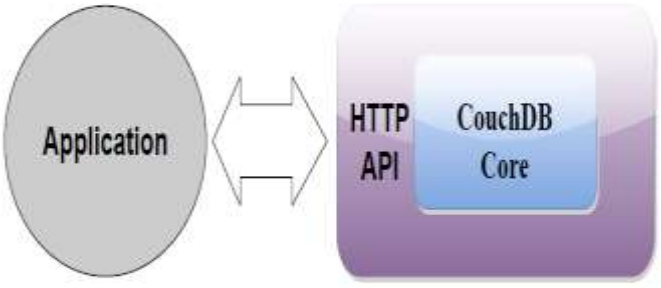

Fig. 2 CouchDB on Single Machine

The main components of CouchDB are B-tree and MapReduce for querying. B-tree is a sorted data structure on which insertion, deletion, and searching can be performed in logarithmic time. CouchDB uses B-tree storage everywhere, also for internal data, and documents.

CouchDB uses views to create relationships among documents and it also provides aggregation with reporting feature. The reason behind using views is that the database works in schema-free manner. [6]

Relational databases sometimes use locking mechanism to maintain the concurrency during transactions. Locking mechanism prevents one user from accessing data while another user is updating the same data at same time. This prevents multiple users from making changes to the same set of data at the same time but if the there are many user using the system concurrently, it becomes common that the database can get stuck in finding out which user should receive the lock and maintaining the lock queue.

CouchDB solves this problem by using Multi-Version Concurrency Control (MVCC) where snapshot of the latest version of the database is provided to each user. The changes are seen by other users only when the transaction is committed successfully by a user. There are many modern databases (Oracle, MySQL (with InnoDB engine) and SQL Server 2005 and later versions) have started using MVCC rather locking mechanism.[7]

1.1.3 Apache Cassandra: Cassandra a type of NoSQL database which is massively scalable. As technical aspects Cassandra can be found at companies recognized for their ability to manage big data effectively –Amazon, Google and Facebook.

In today's environment Cassandra is used for modern businesses to handle their critical data infrastructure, and known for being the solution for the technical professionals when they require a NoSQL database that gives high performance at massive scale, that never degrades the performance of operations. Cassandra is used for unstructured data as big data application, which are mostly used across nearly every industry.

This model is a partitioned in row store with consistency. [8] These are arranged into tables, primary key is assigned always as first component and rows are clustered in the remaining fields of the key. Columns are indexed through primary key.

Tables may be structured, deleted, and modified at runtime without blocking updates and queries. [8]

Joins and sub queries are not supported by the Cassandra except for batch analysis via Hadoop, rather it performs denormalization through features like collections.

1.2 JSON: JavaScript Object Notations (JSON) is an open standard format which gives a way to transmit data objects consisting of attribute-value pairs in form of human readable text. It is used as an alternative to Extensible Markup Language (XML).It is a way of sending and receiving data between web applications and servers. JSON was first specified by Douglas Crockford.
The data is stored in files with .json extension. [8]There is a rise of websites which are able to load data quickly and asynchronously. These sites are powered by AJAX. These sites work without delaying page rendering process. This allows switching up the content of particular element within out layouts without the need of refreshing the page.
In recent years the increase in popularity of social media, may websites rely on the content provided by Facbook, Twiter, Flickr and others. These websites provide RSS feed, which are not easy to work with AJAX. JSON solves the cross-domain issue. This capability of JSON makes it as incredibly useful as it opens up a lot of doors.

```
{
c_id:"1",
c_name:"Ashu",
value:700,
status:"1"
}{
c_id:"2",
c_name:"Ankit",
price:800,
status:"1"
}.
```

Above document file is json format and having information of customer. These are separated in row format by using curly braces. It is easy to read and understand.

1.3 MapReduce: MapReduce is programming model for solving problems in parallel manner across huge sets of data using large number of nodes (referred to as cluster).There are two steps involved in this processing. First of all users specify a map function that processes a key/value pair which then generate a set of intermediate key-value pairs. Now the reduce function performs merging of all the values associated with the same key. [10]
These Programs are written in functional style of programming which are executed in parallel manner on cluster of nodes. Partitioning of input data, order of execution across several nodes and handling failures are done by run time system. That allows developers to program it without having experience in distributed and parallel systems.

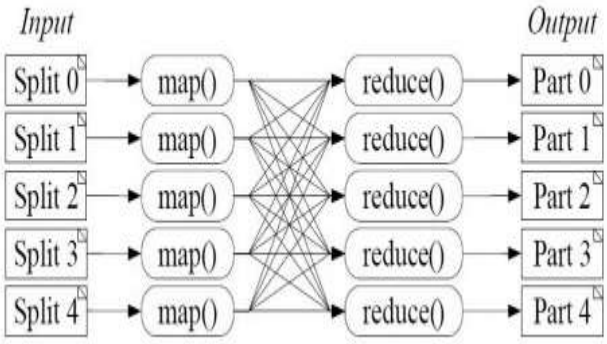

Fig 3. MapReduce Architecture

The MapReduce function is carried out in two basic operations: Map and Reduce. The Map function reads sets of data and performs computations on it. Then the resulted intermediate (key, value) pairs are further passed to Reduce function. Reduce function groups all the values for a each unique key generated by Map function .The keys are presented in sorted manner.
To understand the functioning, the example MapReduce function is shown above. This example counts the sum of occurrences of each word in large set of documents. Map function reads the data of the document and parses out the words. In map step for each word, (key, value) pair is generated i.e. (word,1). Here word is a key and value 1 shows number of occurrence of word is one in the document. [10]
Then the keys/pairs are sorted according to the keys and reduce operation is called for each unique key. Reduce function merges all the values of each unique word in collection of documents. This shows the total count of occurrence of each word in all documents.

1.3.1 MapReduce using MongoDB

Consider the following document which is storing the information of customer. The document stores c_id, name, price and status of the customer.

```
{
c_id:"2",
c_name:"Ravi",
price:652,
status:"1"
}
{
c_id:"5",
c_name:"Hari",
price:522,
status:"1"
}
```

Now, we will execute a mapReduce query on document to select all the customer's information who has active status, group them on the basis of c_id and then calculate the sum of values of data by each user using the following code:

```
db.data.mapReduce
(
 function()
 {
 emit (this.c_id, this.value);
 },
```

```
function(key, values) {return Array.sum(values)},
{
query: { status:"1"},
out:"sum"
}
)
```

This operation will return the following output.

1.3.2 MapReduce with CouchDB

For Relational Databases if the data is structured then we can query anything we want. The problem arises when the data is unstructured. Then we need a different approach to solve this problem. CouchDB uses MapReduce approach to solve this issue. MapReduce is a programming model which performs computations on data in basically two steps: map and reduce. In CouchDB the combination of map and reduce is called a view. These functions make CouchDB very flexible: MapReduce can adapt to variety of documents. [10]

Map function is applied to all documents and then it has emit function which generates zero or more key-value pairs (view rows).Views can be generated in parallel because the map function does not depend on outside information from the document. Views are stored as rows in sorted manner by keys in B-Tree. It makes the data retrieval efficient. The goal of writing map function should be to build an index that stores related data records by using nearby keys.

Map function has one parameter "doc", which refers to a document from the database. Map function contains emit function which can be called any number of time. The result from the emit function is stored in the B-Tree like documents but in their own files. Map Function can return keys or list of several keys.

After map function returns the group of key-value pairs, a series of reduce functions are called for each key. These functions are executed on sorted rows emitted by map function. CouchDB functions takes advantage of storing data in B-tree like documents. The view result is achieved by preorder traversal of the tree. The reduce function are computed from the leaf nodes to the root. So the result of this traversal is cache which can be updated incrementally as data changes. In this procedure first the map results are recalculated and then reduce function is operated. Caching of reduce results are done in the intermediate nodes of the tree.

2. LITERATURE SURVEY

In 2013, Sanobar Khan and Prof.Vanita Mane, in their research paper with title "SQL Support over MongoDB using Metadata" have given the comparison between RDBMS and MongoDB. From their research they found that still RBMS has its own significance but not best for large amount of data. MongoDB is better than RDBMS it is very easy to use and give best performance at large scale. they found that if your database is having large datasets then choose MongoDB for better performance [9].

3. PROPOSED APPROACH

Mongodb, CouchDB and Cassandra are NoSQL databases which are used when data is huge. Here JSON file is used to store large amount of data. On which Mongodb operations are performed such as MapReduce with several conditions.

There are for collections created in Mongodb. First collection contains 50k records, second collection contains 100k records, third collection contains 500k reocrds and fourth collection contains the 1000k records in it.

CouchDB and Cassandra follow the same scenario that same amount of records as Mongodb has.

Ubuntu platform is used to perform the operations on Cassandra, Mongodb and CouchDB. These databases provide the high speed and high throughput as compare to relational databases.

4. EXPERIMENTAL SETUP

In this research, all the tests are performed under following specifications:

1) **Host System:** Intel i5 core processor with 6 GB RAM and 1000 GB Hard disk.
2) **Operating System:** Ubuntu
3) **Mongo DB**
4) **CouchDB**
5) **Cassandra**

a) **Execution Time:** Execution time can be defined in terms of time consumed by an algorithm in order to solve a problem using processor p.

5. RESULTS AND ANALYSIS

Experiemnt-1: This experiment finds out the find out the number of customers and group by customer id. This operation can be performed in MongoDB, , Cassandra and CouchDB databases.

```
db.data100k.aggregate
(
[{
$group : {_id : "$cust_id", count : {$sum : 1}}
}]
)
```

This operation can be perform in Couchdb as following query

```
// Map Function
        function(doc) {
        for (var x = 0, len = doc.People.length; x <len;
        x++)
          {
        emit(doc.People[x].cust_id, 1);
          }
        }
// Reduce Function
function(keys, values, rereduce) { return sum(values);}
```

In case of Cassandra the following code will perform this operation.

Select cust_id from data100k group by cust_id

Table: 1

DOI: 10.18535/ijecs/v5i1.29

EXECUTION TIME FOR MONGODB, CASSANDRA AND COUCHDB FOR EXPERIMENT-1

Data Records	Mongodb Time	Couchdb Time	Cassandra Time
50k	0.523	0.75	0.253
100k	1.457	2.36	0.759
500k	4.933	5.726	2.96
1000k	7.653	10.559	6.42

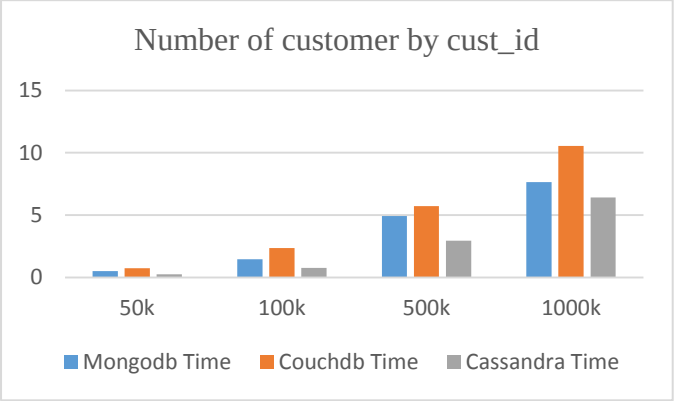

Fig 4. Execution Time for MongoDB, Cassandra & Couchdb for Data Retrieval

From the figure 4 it is clear that execution time taken by Cassandra is better than CouchDB and MongoDB for different numbers of records. As the number of records increases performance of Cassandra is also increased for the data retrieval operation in comparison to CouchDB and MongoDB.

Experiment-2:- This experiment finds out the sum of prices of all customers. We can use mapReduce function on document to select all the customers and find out the sum of prices of each user.
This operation in MongoDB can be performed using following code.

```
db.posts.mapReduce(
function() { emit(this.cust_id, this.price); },
function(key, values) {return Array.sum(values)},
    {
out:"total_price"
    }
)
```

In CouchDB code as follows

```
// Map Function
function(doc) {
for (var x = 0, len = doc.People.length; x <len; x++)
{
emit(doc._id, doc.People[x].price);
}
}
// Reduce Function
function(keys, values, rereduce) {
return sum(values);
}
```

In case of Cassandra the following code will perform this operation.
Select SUM(price) from data50k

Table: 2
EXECUTION TIME FOR MONGODB, CASSANDRA AND COUCHDB FOR EXPERIMENT-2

Data Records	Mongodb Time	Couchdb Time	Cassandra Time
50k	0.365	0.533	0.374
100k	0.844	1.87	0.644
500k	1.475	2.1	0.866
1000k	2.632	3.88	1.398

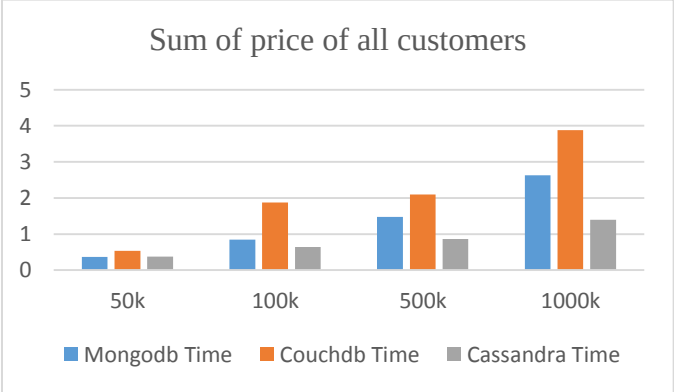

Fig 5. Execution Time for MongoDB, Cassandra & CouchDB for Experiment-2

From the figure 5 it is easy to analysis execution time taken by Cassandra is better than CouchDB and MongoDB for different numbers of records. As the number of records increases performance of Cassandra is also increased for the data operation in comparison to CouchDB and MongoDB.

Experiment-3:- This experiment finds out the sum of prices by customers which are grouped by customer id. This operation can be performed in MongoDB, Cassandra and CouchDB databases. In this experiment MapReduce function is used to achieve the task.

This operation in MongoDB can be performed using following code.

```
db.posts.mapReduce(
function() { emit(this.cust_id, this.price); },
function(key, values) {return Array.sum(values)},
    {
out:"total_price"
    }
)
```

in CouchDB CQL code will be as

```
// Map Function
function(doc) {
for (var x = 0, len = doc.People.length; x <len; x++)
  {
emit(doc._id, doc.People[x].price);
  }
}
// Reduce Function
function(keys, values, rereduce) {
```

return sum(values);}

In case of Cassandra the following code will perform this operation.
Select cust_id, SUM(price) from data50k group by cust_id

Table: 3
EXECUTION TIME FOR MONGODB, CASSANDRA AND COUCHDB FOR EXPERIMENT-3

Data Records	Mongodb Time	Couchdb Time	Cassandra Time
50k	0.307	0.41	0.233
100k	0.536	0.887	0.256
500k	0.863	1.741	0.376
1000k	2.44	3.86	0.985

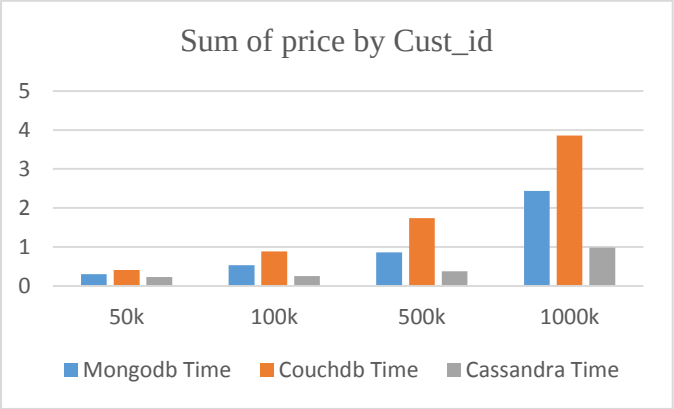

Fig 6. Execution Time for MongoDB, Cassandra & CouchDB for Experiment-3

From the figure 6 we can analysis that for less number of records the execution time for CouchDB and MongoDB is not very different but as the number of records increases performance of Cassandra is increased for the data extraction in comparison to MongoDB and Couchdb.

Experiment-4:- This experiment finds out the count of all customers in the database. This operation can be performed in MongoDB, Cassandra and CouchDB databases. MapReduce function is used to achieve this task. We can use mapReduce function on collection of documents to select all the customers and count the total number of customers.

This operation in MongoDB can be performed using following code.

```
db.posts.mapReduce(
function() { emit(this.cust_id,1); },
function(key, values) {return Array.sum(values)},
    {
out:"total_customers"
    }
)
```

in CouchDB CQL code will be as

```
// Map Fucntion
function(doc) {
for (var x = 0, len = doc.People.length; x <len; x++)
  {
emit(doc._id, 1);
  }
}
// Reduce Function
function(keys, values, rereduce) {
return sum(values);
}
```

In case of Cassandra the following code will perform this operation.
Select count(cust_id) from data50k

Table: 4
EXECUTION TIME FOR MONGODB, CASSANDRA AND COUCHDB FOR EXPERIMENT-4

Data Records	Mongodb Time	Couchdb Time	Cassandra Time
50k	0.354	0.477	0.241
100k	0.647	0.841	0.375
500k	0.745	1.968	0.522
1000k	1.863	2.74	0.842

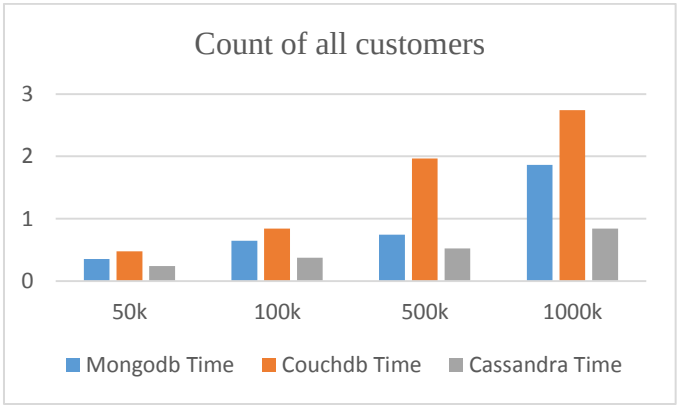

Fig 7. Execution Time for MongoDB, Cassandra & CouchDB for Data Retrieval

From the fig 7 we can analysis that execution time taken by Cassandra is better than CouchDB and MongoDB for different numbers of records. As the number of records increases

performance of Cassandra is also increased for the data retrieval operation in comparison to CouchDB and MongoDB.

6. CONCLUSION

As the number of records in database increases, the difference between the execution time taken by Cassandra for the computation of different database operations is better in comparison to CouchDB & MongoDB.

For finding the number of customer which can be seen as data retrieval operation, the performance of Cassandra is about 35% better in comparison with CouchDB & MongoDB, for the different numbers of records.

For finding the sum of prices of all customers which can be seen as performing total operation on the data Cassandra seems better in comparison with CouchDB & MongoDB significantly. Collectively we can say that for all database operations Cassandra is much better than CouchDB & MongoDB, whether the number of records are less or large.

7. FUTURE SCOPE

The present and future scope of NoSQL Databases are bright. There are many opportunities and big challenges which need to be overcome.

In future we can perform the analysis on different databases like graph databases and key value databases.

8. REFERENCES

[1] http://nosql-database.org/

[2] Dominik Bruhn, "Comparison of Distribution Technologies in Different NoSQL Database Systems".Available:http://files.dbruhn.de/pub/Comparison_of_Distribution_Technologies_in_Different_NoSQL_Database_Systems-SA_Dominik_Bruhn.pdf

[3] NoSQL databases by Christof Strauch. Available:http://www.christof-strauch.de/nosqldbs.pdf

[4] Hsinchun Chen and Roger H. L. Chiang, "Business Intelligence and Analytics: From Big Data To Big Impact", MIS Quarterly Vol. 36 No. 4/December 2012 BIGDATA MongoDB, http://www.mongodb.org big data

[5] http://couchdb.apache.org/

[6] Taylor, R. C. (2010) An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics, Proc. 11th Annual Bioinformatics Open Source Conference (BOSC).

[7] Oliver Schmitt and Tim A. Majchrzak of University of Münster, Germany, "Using Document-Based Databases for Medical Information Systems in Unreliable Environments".

[8] Eben Hewitt, Apache cassandra project chair "cassandra the definitive guide" Published by O'Reilly Media November 2010.

[9] Sanobar Khan - Department of Computer Engineering, RAIT, and Prof.Vanita Mane - Department of Computer Engineering, RAIT "SQL Support over MongoDB using Metadata ",International Journal of Scientific and Research Publications, Volume 3, Issue 10, October 2013.

[10] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters", Commune. ACM, Pages:107-113, 2008.

[11] Kai Orend, "Analysis and Classification of NoSQL Databases and Evaluation of their Ability to Replace an Object-relational Persistent Layer".

[12] Massimo Carro, NoSQL Databases, http://arxiv.org/abs/1401.2101. CoRR, 2014.

[13] Robin Henricsson, Blekinge Institute of Technology, in 2011