Data Storage and Management Project

Technical Report · February 2019

DOI: 10.13140/RG.2.2.27354.18880

1 author:

Akshay Mungekar
National College of Ireland
4 PUBLICATIONS 0 CITATIONS

SEE PROFILE

Data Storage and Management – Project B

Name : Akshay Mungekar

Student ID : x18103952

EFFICIENCY ASSESSMENT OF HBASE AND MONGODB USING YCSB

Abstract —
The research is performed on the two NoSQL(Not only Structured Query Language) databases, HBase and MongoDB, using an open-source capability test tool called as YCSB (Yahoo Cloud Serving Benchmark).The purpose of this research paper is to perform an efficiency test on two of the mentioned NoSQL database systems. The differentiation between the two databases will be based on the results or output obtained from the YCSB tests. As these NoSQL databases are supposed to offer increased performance and availability in comparison to the traditional databases, the results will allow us to evident this claims. Along with that, it will also allow to distinguish between the two chosen NoSQL databases on the basis of the performance evaluation parameters used in the tests. The evaluation parameters take a note of the results from the CRUD operations to be executed on the selected workloads, for both the databases. On the basis of the obtained output, the efficiency of the compared databases can be interpreted. This will further help to attain any performance related issues appearing in any of the evaluated databases.

Index Terms- ycsb, hbase, mongodb, testharness, workload, testharness,

I. INTRODUCTION

In this research paper, the area of discussion is based on the database systems and its performance measures. Also, sufficient knowledge on the evolution of different forms of databases along with the characteristics, significance and usability of the two chosen NoSQL databases have been represented. Before understanding these database concepts, the term data or more correctly big data should be known in this context. Therefore, the title Big Data is expressed as the computing potential of the traditional database structures. The data being massive, fails to adapt to this traditional database structures. Thus, a new form of database structure was created in the year 1998; which is called as NoSQL database. There are four major classifications of it. They are Key-Value Store, Wide Column Store, Document Store and Graph Store. Also to be known, there are a number of database programs being developed using this architecture. Some of its examples are the HBase, MongoDB, Cassandra, Redis, Neo4j and many more. For this research, the tests have been carried out on the HBase and MongoDB database systems (MATEI, 2014). The benchmark tool used to perform the tests is YCSB. It is nothing but a Yahoo Cloud Utility Benchmark Client. The principal aim of this tool is flexibility and scalability because the performance of the newly introduced cloud based databases can also be measured. Another benefit of using this tool is, it is capable to run the same workloads against the each of the databases and also can have multiple databases on the same hardware setup. Further, this YCSB framework has a primary collection of parameters that are utilized to evaluate the systems efficiency. The reason that makes it more usable is that, YCSB allows the users to make changes to its already existing packages as well as adding some of their own packages (Rohini Gaikwad, 2015). The motivation behind this paper is to examine and contrast a few NoSQL databases to find the different usefulness and insights from each one of them; and to decide which of them is more favourable for the applied workloads. The procedure of examination and correlation will include the introduction and a short explanation of the selected database systems, testing tool and finally the comparison between the databases based on the test output (MATEI, 2014). The modules that will be discussed in this paper is as follows,

1. Key Characteristics of Chosen Data Storage Management Systems
2. Database Architectures
3. Chosen Area from Requirement C section
4. Learning from Literature Survey
5. Performance Test Plan
6. Evaluation and Results
7. Conclusion and discussion

II. Key Characteristics of Chosen Data Storage Management Systems

i. The principal characteristics of the HBase database management system (Guru99, 2018) (jaiswal, 2018) :

- HBase is a column-oriented NoSQL database application.
- It is meant to work on low latency tasks.
- It is largely used to execute random read/write tasks.
- It preserves vast measure of information in tabular form.
- It performs parallel data processing through MapReduce which can also be backed-up.
- It allows for linear and modular expansion of data
- It behaves more stable with read/write operations.
- Automated configuration and horizontal partitioning (sharding) of large tables.
- Aborted workload operation support for the data in the DataNode
- Java based APIs are simple to use with it.
- It can handle structured as well as semi-structured data.
- Real-time data processing is accomplished using the bloom filters and block cache.

ii. The principal characteristics of the MongoDB database management system (Guru99, 2018):

- MongoDB is a document-oriented NoSQL database application.
- It provides a foremost filtration option to filter based on field, regular expressions.
- Additionally, it also provides indexing of the field.
- It is possible to replicate the data to achieve high availability.
- It also allows horizontal partitioning (sharding) of large data on various instances.
- It is capable to implement aggregation of data.
- It reduces stack complexity for large data files.
- It also has a built-in web tool that keeps a check of the data and its performance, and also deals with the instance related issues.
- It hands a multiple storage engine feature, that manages most of the memory management tasks.
- Moreover it is easy to configure, provides much faster operations and good for hierarchical data storage.

III. Database Architectures

i. HBase architecture :

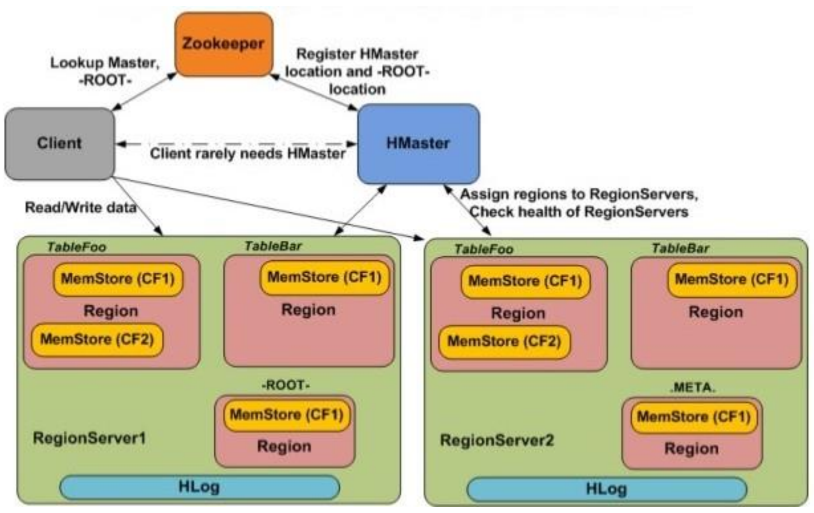


Figure 1 : HBase architecture (Dezyre, n.d.)

Apache HBase, developed by apache software foundation, was released in March 28 2018, is an open-source, distributed database system. It is developed upon the HDFS and thus offers high performance in terms of parameters such as read, write tasks, high throughput and low latency (Anon., 2008). The components of the HBase architecture are listed which can be seen in the above architectural Figure 1. The entire work flow of the HBase is mainly based on these components. To begin with the flow, it is always worth to mention that this database system is meant to achieve crucial goals like low latency, high throughput and other such performance measures. Also, the tables that stores over here are randomly spread in case of over populating of the tables. These horizontal spread is taken care of by the region component. It has a master node known as HMaster along with multiple slaves known as the region servers. HMaster takes care of the request received from the client and later transfers it to the appropriate region server.

Below are the list and explanation of all the components attached with the HBase architecture,

i. HMaster :

HMaster handles a collection of Region Server which resides on DataNode. Let us understand how HMaster does that. It manages the underlying region servers known as slaves. These slaves occupy the memory space of the DataNodes (Dezyre, n.d.). It handles the create and delete table tasks and then allocates those areas to the slave servers on the initial occasions as well as while recovery and load stabilizing tasks. It controls the region servers in complete synchronization. It is responsible to examine the instances of the region server and to handle the situations such as downtime of the servers. It also has a duty to observe the instances of the slave servers.

ii. ZooKeeper :

Zookeeper can be termed as an organizer within the HBase distributed system. The current phase of the server is looked over by it. This is done with the use of the sessions. Every Region

Server along with HMaster Server sends continuous heartbeat at regular interval to Zookeeper and it checks which server is alive and available as mentioned in above image. It also provides server failure notifications so that, recovery measures can be executed.It also sends the notification about theserver inactiveness to the zookeeper to quickly carry out necessary steps.

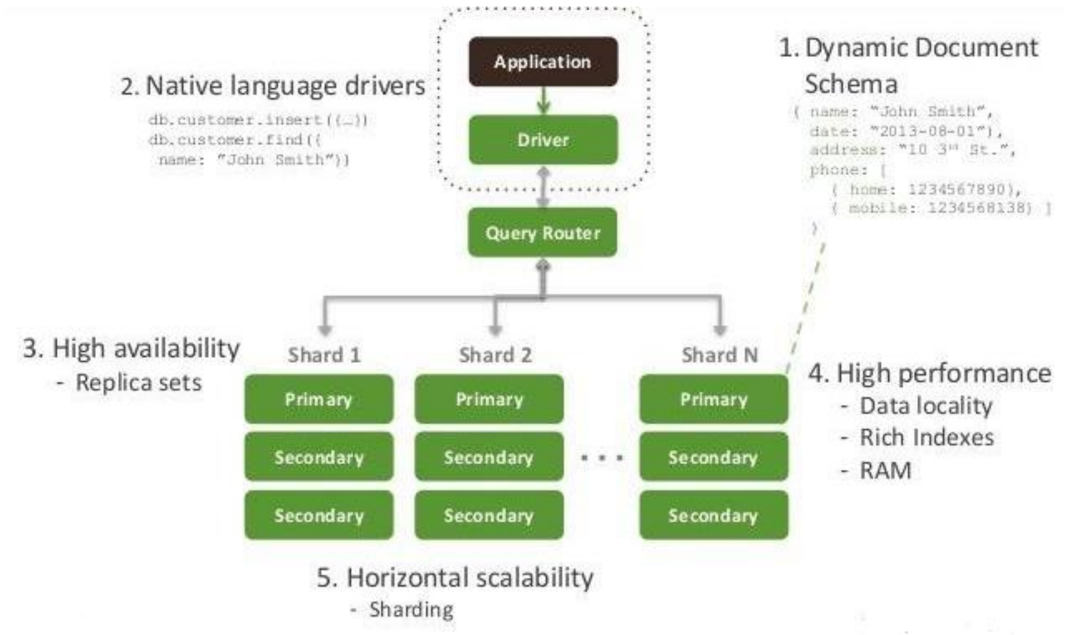
There is also a provision for a backup server if any required. The available Hmaster transmits heartbeats to the Zookeeper. At the same time, the unavailable HMaster waits for the acknowledgement from the active one. If the wait is longer then the session is terminated and the unavailable HMaster becomes available.

iii. Region Server

As mentioned, these are the worker or slave nodes. These nodes are responsible to serve the CRUD requests. The slave server tasks are meant to be active on every node in the Hadoop groups. More specifically, they are meant to run on the DataNode of the HDFS. It consists of the below listed components, (Brain4ce Education Solutions, 2014)

- Block Cache : This cache is used to store the read information.
- MemStore : This cache is used to store the write information of the data to be written to the disk.
- Write Ahead Log (WAL) : The newly added data that is yet to exist in the permanent storage is contained inside this file.
- HFile : The appropriate key-value pairs are obtained from this storage file.

ii. MongoDB architecture :



Application
Driver
Query Router
1. Dynamic Document Schema
{ name: "John Smith", date: "2013-08-01"), address: "10 3rd St.", phone: [{ home: 1234567890), { mobile: 1234568138)] }
2. Native language drivers
db.customer.insert({…})
db.customer.find({ name: "John Smith"})
3. High availability
- Replica sets
Shard 1
Shard 2
Shard N
Primary
Secondary
Secondary
4. High performance
- Data locality
- Rich Indexes
- RAM
5. Horizontal scalability
- Sharding

Figure 2 : MongoDB architecture (researchgate, 2018))

MongoDB, initially developed in 2007 by Eliot and Dwight for 10gen, is document based database system that can run on multiple different platforms (wikipedia, 2009). It falls under the category of NoSQL database.
One of the major reason for its creation is the management of huge amount of data. The data is managed in the form of a document and it contains various fields. The length of every document can differ a lot when compared with the other documents. The structure of such documents solely depends on how the developers program it to be like, as there is not any predefined schema to create develop it (Guru99, 2018).
It provides support for Ad hoc queries, indexing, workload balance and replication.

To model the data, always consider the demands of the application along with the type of the data. After this, also make sure about the document type to be considered. In case of heavy data to be modelled, then it is anytime important to index the data to get the full efficiency out of it. Adding to this, it is also necessary to have a frequent check on the CRUD operations. If found any inconsistencies, then utilize the sharding feature to increase the efficiency. (Guru99, 2018)
Apart from this, MongoDB has several useful drivers incorporated such as the java driver and also clients like java client.

Below are the lists and explanation of each of the components,

i. Configuration Servers :

   The metadata about the sharded cluster is preserved by the Config Server. They are also termed as the particular mongod instances (wideskills, 2015).
   The method used by these servers is named as two phase commit. This method performs a check for constancy and accuracy.
   Also, it is responsible to deploy a shared cluster and modifying the sub-data.There is a cluster known as production cluster, which is scaled horizontally. This cluster includes more three config servers. The cluster shouldn’t contain less than three such servers, otherwise it would result in a single point of failure. Adding to that, if the any of the config server is unavailable at the moment then it is not possible to access the whole cluster. It is also important to know, the cluster turns inactive if the config server is not able to retrieve some of the information. Each sharded cluster has an attached config server. A confid database is used by the server to store the supportive data. Moreover, the data is written to the config server by MongoDB while developing splits in the chunks present and also whil moving chunks between shards. On the other side, the data is read from the config server when the fresh mongo instances begin for the first time or even if an inactive mongo restarts.

ii. Query Routers :

   MongoDB consists of a minimum one mongos instance that further acts router.
   This query router sharding is such that it is transparent to the applications irrespective of the number of shards.
   In this, there is generally one mongos instance assigned for every application server.

iii. Shards :

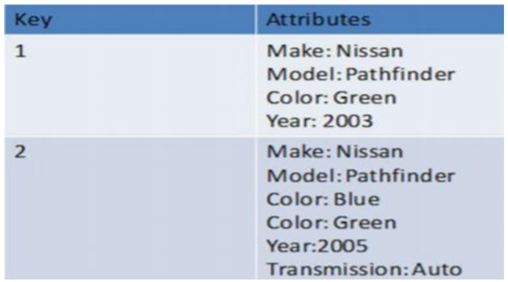
   Shards are nothing but a formation of 2 or more replica groups (wideskills, 2015).
   With this, the processing capabilities are extended to a huge margin though MongoDB without any frequent downtimes and also without any data corruption.

IV. Storage and retrieval of Structured, Semi-Structured and Unstructured Data

Figure: Storage and retrieval of Structured, Semi-Structured and Unstructured Data (Foote, 2018)

(Jagdev Bhogal, 2015) As discussed above, there has been a rapid increase in the amount of data. Thus, to counter these concerns, a new database technology known as the NoSQL was introduced. One of the vital reasons to utilize this technology is the support for wide variety of storing data it equips that can even be either structured, semi-structured or unstructured. Other important reason that constitutes a crucial hand for handling such types of data is that it has Schema less design. The social media based organizations mainly use it on a large scale. The storage mechanism works in many different ways to manage the heavy burden of multiple types of big data. It performs scaling of data by distributing the load between multiple db servers. There is also a method known as record tracking to optimize preserving big data.The data storage and retrieval mechanism in NoSql works in various possible ways and has many different approaches.

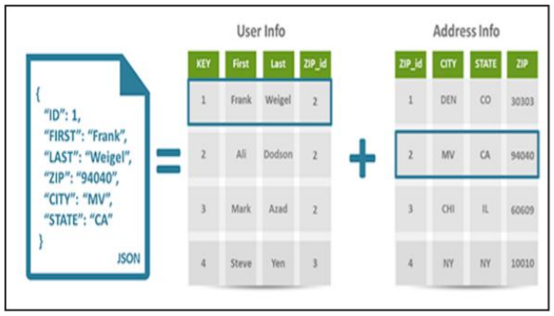
One of them is the key-value approach. It makes use of the hash table which allows to fetch the required data based on the unique key and pointer. This approach is mainly used to retrieve unstructured as well as semi-structured data, as such type of data is difficult to be retrieved.
HBase, a NoSQL database approach uses this form of database approach to store big data. Due to easy to learn design of this database system, it is highly possible to quickly scale and retrieve the values required by the application tasks such as fetching the stored specific customer names. Some big names like Amazon uses DynamoDB, a key-value based NOSQL database approach, to manage its website. Thus, this allows for faster and efficient governing of the data.



Key	Attributes
1	Make: Nissan Model: Pathfinder Color: Green Year: 2003
2	Make: Nissan Model: Pathfinder Color: Blue Color: Green Year:2005 Transmission: Auto

Figure : Key-Value data storage (Jagdev Bhogal, 2015)

The document database is another approach of the NoSQL database. This approach considers the whole document of the structured datasets. Generally, every record is preserved in one document along with all of its data to allow easy access to the data. This technique is mainly used to handle the unstructured data. This is also useful to manage the huge collections of document data like text documents, emails etc. **MongoDB**, a NoSQL database system, is built upon this approach. There are many more approaches like the Column-oriented and the Graph database.



```
{
  "ID": 1,
  "FIRST": "Frank",
  "LAST": "Weigel",
  "ZIP": "94040",
  "CITY": "MV",
  "STATE": "CA"
}
```

User Info

KEY	First	Last	ZIP_id
1	Frank	Weigel	2
2	Ali	Dodson	2
3	Mark	Azad	2
4	Steve	Yen	3

Address Info

ZIP_id	CITY	STATE	ZIP
1	DEN	CO	30303
2	MV	CA	94040
3	CHI	IL	60609
4	NY	NY	10010

Figure : Document data storage (Jagdev Bhogal, 2015)

From the above discussed methods for storage and retrieval of data, the study of how Hbased uses the table based approach has been described. The structure of Hbase storage and retrieval includes the following components in the names of attribute dictionary, inverted list and the database itself (Pijin Gong, 2017). In the attribute dictionary, the object attribute acts as an index tracker for the application system. This dictionary is nothing but a cluster of all the object based attributes. The data about the attributes is contained within this dictionary. In the inverted list, it holds a collection of index tracks for the objects. With this tracks, it is easy to track the objects holding attribute. These attributes are stored in a sequential manner in a file which is termed as inverted file. This file further holds the inverted index.This defines the basic flow for storage of any complex data structures. Moving on, the next is the retrieval of the stored data. This process involves the server that requests the client to sends the input attribute value via a relevant network. Further, the server looks for the value of the attribute from the property dictionary.
Thereafter, the server searches the attributes from the above list, then reads the object from the Hbase with the help of unique indexes for the records in the list.

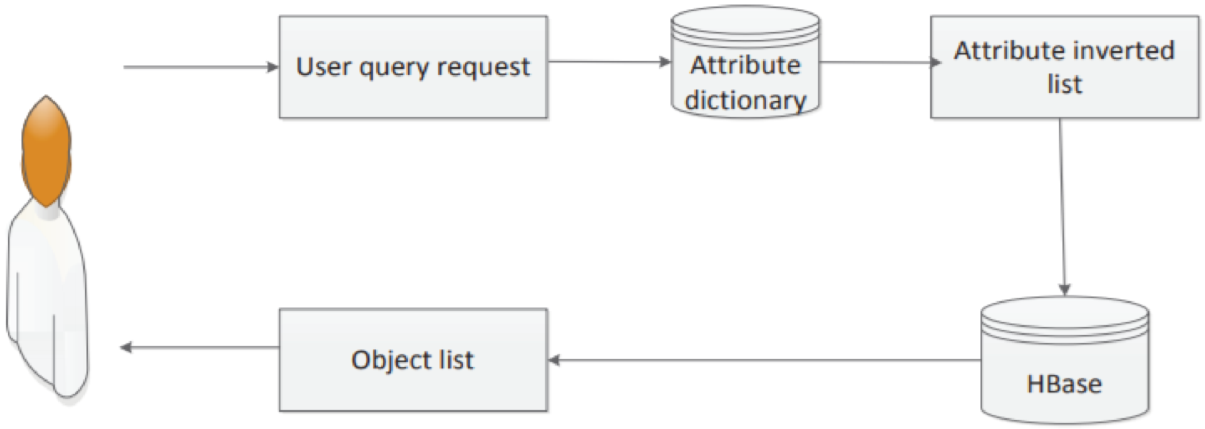


Figure : Hbase storage and retrieval block diagram (Pijin Gong, 2017)

Another important and useful approach is the MongoDB database and the study of how MongoDB uses the document based approach which has been previously described. The structure of MongoDB storage and retrieval includes the several components such as the config servers. Many algorithms like the Consistent hashing algorithm can be used with MongoDB to store and retrieve large amounts of data. It also follows a similar key value based trend. It is good enough to deal with server issues like inactiveness or the network related issues. On the other hand, it is not complicate to track a record value as the servers are already indexed from 0 to server count – 1. Therefore, by hashing the values key % s results out the appropriate linked server. Whenever the server is inaccessible, then it is a matter of concern as the server is unable to fill in the hash memory. In such cases, discard the caches on the servers and begin fresh by re-indexing them. To overcome such issue with the retrieval of the data, hash value is created for each of the server and is similar to positioning it on a point of circle (formed by wrapping the hash memory). Further if the key is to be searched for, then the hash is performed again that directs to the point of the same circle. Afterwards, it then revolves clockwise around the circle till a server is hit. If no server is hit then it considers the first server. Another issue then arose due to heavy load on the first server. This issue can be tackled by just adding every server to the ring numerous times in variable places. To be more specific, a separate replica count is required to achieve it.

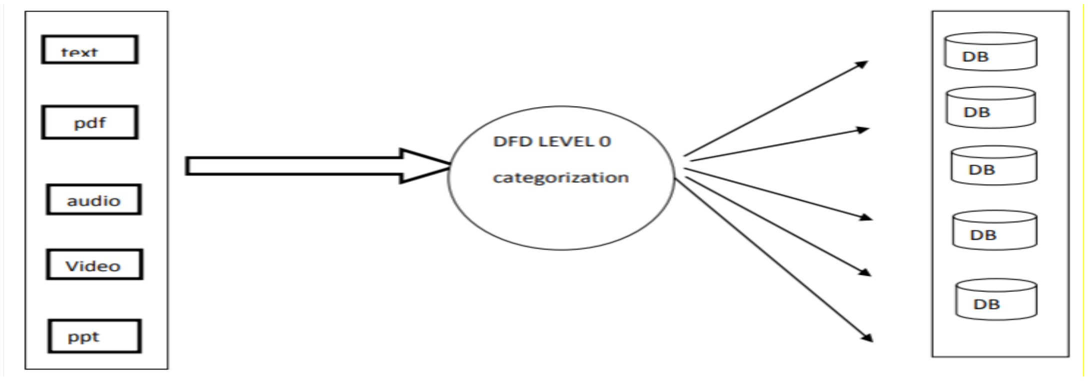


Figure : MongoDB storage and retrieval block diagram (Raj, 2015)

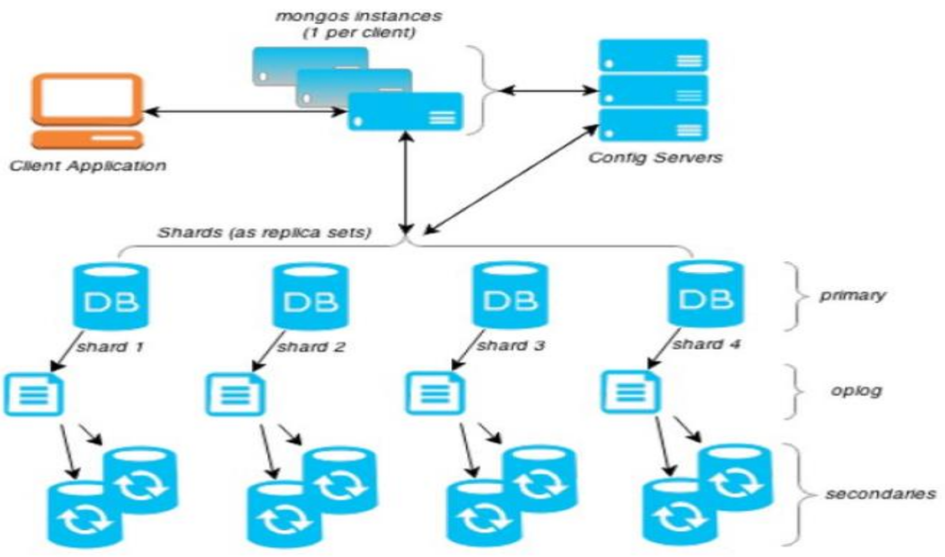


Figure : MongoDB storage and retrieval component diagram (researchgate, 2015)

V. Learning from Literature Survey

1. (Rohini Gaikwad, 2015)The research paper is based on the study of YCSB tool that is used to measure the performance of NoSQL database.
   This paper focuses on findings resulted by testing the various NOSQL database performance, based on several performance measures.
   It also specifies and explains the tool used to evaluate the performance of the database.
   It also details about the workload in terms of the operation parameters and the different forms of workloads.
   Also, the authors described about the different versions of YCSB
   The researchers concluded stating that the YCSB is the best performance measurement tool for NoSQL database and also added certain suggestions to make YCSB problem-free and more usable.

2. (EnqingTang, 2016) +In this research, the authors have discussed on comparing five NoSQL databases (Redis, MongoDB, Couchbase, Cassandra, HBase) based on certain performance criteria.
   They also state that the comparison would be carried out based on the experimental results obtained using the YCSB evaluation tool.
   They specify the aim of the paper is to test the NoSQL performance.
   Further, they also talked about the operations and workloads used to perform the tests.
   Later, they highlighted the experimental results as per the requirement.
   They also talked about certain improvements needed in terms of load balancing and efficiency.
   The researchers conclude the paper stating that Redis performs extremely well as compared to the remaining four databases.

3. (Yusuf Abubakar, 2014)In this paper, the researchers have talked about the performance measurement of NoSQL databases using YCSB in a resource austere environment.
   The authors focuses on the YCSB framework and the test to be performed on the new generations of NoSQL databases (MongoDB, ElasticSearch, Redis, OrientDB).
   They also specify the hardware details about the experiments to be carried out.
   The paper mainly targets on evaluating the performance of the four NoSQL databases on a cloud platform.
   All the details about the experiment and its results are represented in a detailed manner.
   The researchers conclude stating the advantages and importance of the benchmark tests to be executed.

4. (George Seriatos, 2016)The research paper talks about the comparison of database and workloads in the cloud environments.
   The researchers specify the goal in terms of behavioural study of the several NoSQL databases (MongoDB, Cassandra, HBase) when put on virtual platform.
   They also specify the benchmarking tool, YCSB, used to measure the behaviour
   The paper targets the ability of the databases in terms of workloads, performance measures.
   The authors highlight the YCSB client that is used to launch queries against the database systems on the virtual platform.
   The paper also represents the experimental approach and the results along with its interpretations.
   They also declare MongoDB as the efficient and optimal performer in the test results obtained by YCSB.
   They finally conclude stating the tests performed, allowed to exploit the performance limitations of the different database system operations.

5. (Suman Kashyap, 2013) In this paper, it focuses on the topic Benchmarking and Analysis of NoSQL Technologies.
   The paper discusses about the benchmarking output obtained from the four NoSQL databases (Cassandra, HBase, MongoDB, CouchDB).
   The authors also mention that the results are analysed based on the datastore performance.
   They highlight the tool, YCSB, to test the performance and also the talks about the measures on which the tests will be performed.
   They also declare the new workloads used for the purpose of reliable performance test results and also declare the utilization of the aggregation operations for the same.
   Also, the tradeoffs for the read/write performance have been specified.
   In the end, they conclude stating the ease to create new workloads and the effectiveness of the YCSB tool.

VI. Performance Test Plan

The performance test plan is a strategical approach that specifies the significant factors utilized by us to execute and accomplish the requirements of the process.

i. Performance Test Requirements :

<u>Setup Hadoop environment</u>

- Install a Hadoop instance and execute a sample Map-Reduce activity.
- Configure Hadoop in pseudo-distributed mode. Add the necessary software packages. groups, users. Further, configure SSH and ensure an ssh connection has been established.
- Make sure to disable the IPv6 as it not compatible by Hadoop at present.
- At this point, install and setup Hadoop and also add a symbolic link and provide the read/write permissions to appropriate folders.
- Later, validate the xml files with the correct data in order to facilitate the correct processes in pseudo-distributed mode.
- After this, format the namenode of HDFS and begin all the Hadoop services.

<u>Setup HBase</u>

- Download and extract the Hbase file into appropriate directory.
- Create a symbolic link to that folder.
- Provide read/write permissions to that folder.
- Validate all the xml files accurately in order to facilitate the correct processes in pseudo-distributed mode.
- Further, start Hbase and get in to the Hbase shell.

<u>Setup YCSB</u>

- Download and extract the YCSB file into an appropriate directory.
- Configure the YCSB tool.

<u>Setup MongoDB</u>

- Download the MongoDB binary files and configure it according to the required mode.
- Ensure the session to mongDB is established successfully.

Setup Testharness

- Download and extract the testharness file into an appropriate directory.
- Validate certain configurable files with the accurate data to run the tool disruptly.

ii. Databases, Tools, Workloads and Opcounts undertaken :

- **HBase and MongoDB** are the two NoSQL databases using which the test results are evaluated.
- **YCSB** (Yahoo! Cloud Serving Benchmark) tool used to execute the tests.
- **Testharness** tool is used to automate the entire test operations.
- **Workload a and d** are the two worklo ads on which the test is performed. Workload a supports 50% updates and 50% reads whereas workload d supports 95% reads and 5% inserts.
- 1000000, 250000, 400000, 550000, 700000 are the values for **Opcounts** on which the comparison between the two database systems is carried out.

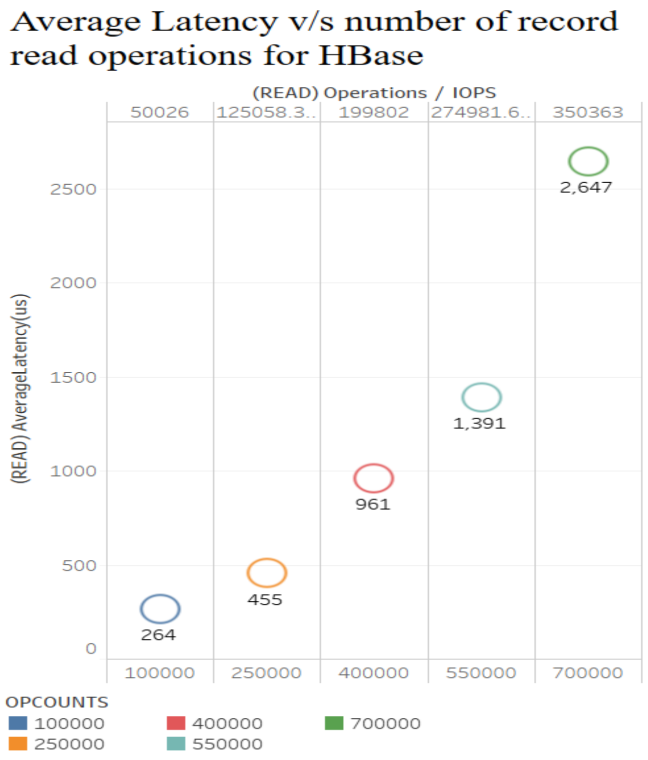
iii. Performance test approach :

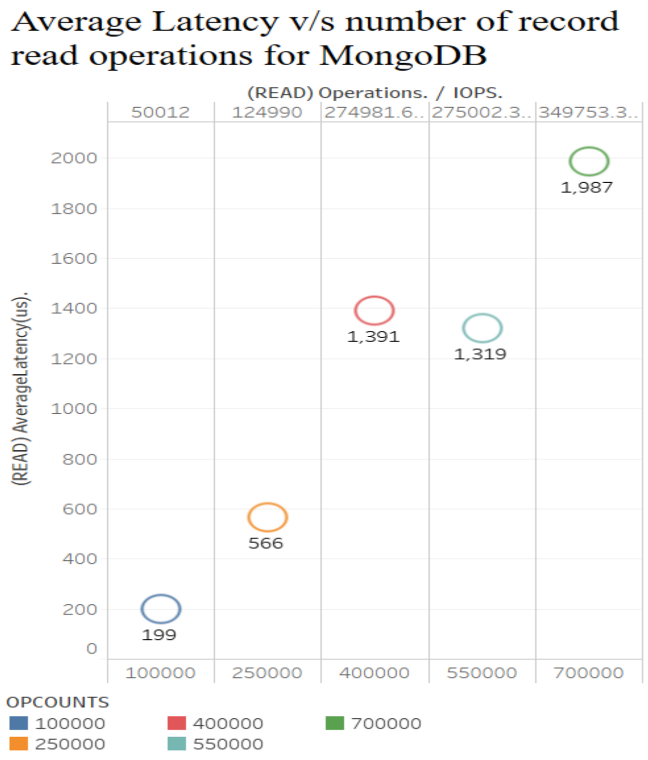
- Firstly, create an instance on the cloud ex. Openstack. Allocate a floating ip to the instance.
- Then open a linux based terminal (Putty) and sign in with the correct details.
- Once done, then follow all the above steps listed in the test requirements section.
- Re-check for configuration files if any inaccuracies observed.
- If all works fine, then execute the YCSB tool thrice that is currently configured for both the databases, HBase and MongoDB, with the pre-defined measurement parameters based on the assigned workloads and the opcounts.
- Later, average the results obtained from three cycles of YCSB.
- Finally, interpret the results with any of the visualization tool (tableau) and then conclude the output.

VII. Evaluation and Results

a) WORKLOAD A :

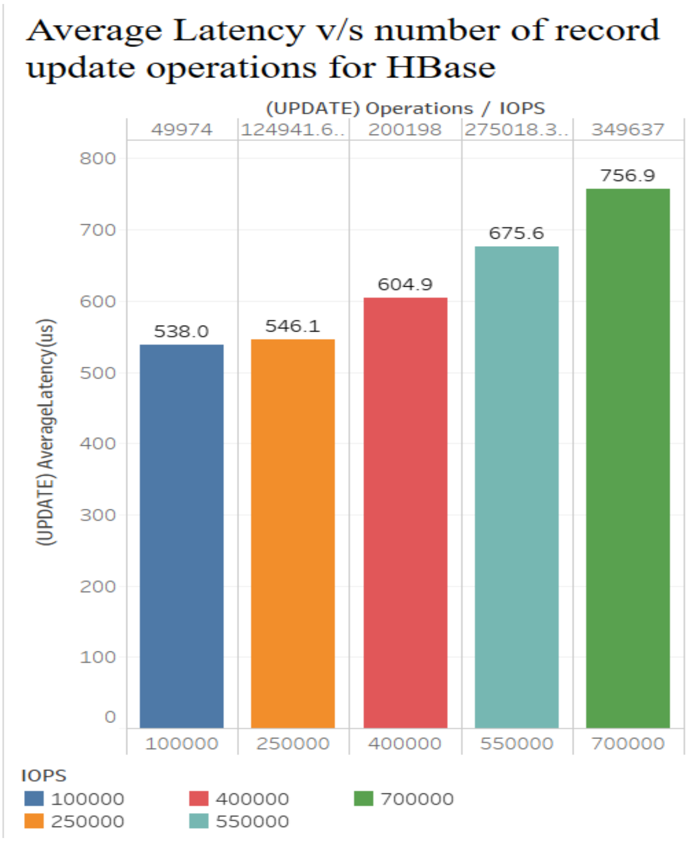
1. A graphical representation of the recorded Average Latency against the number of record read operations for HBase and MongoDB over workload a,

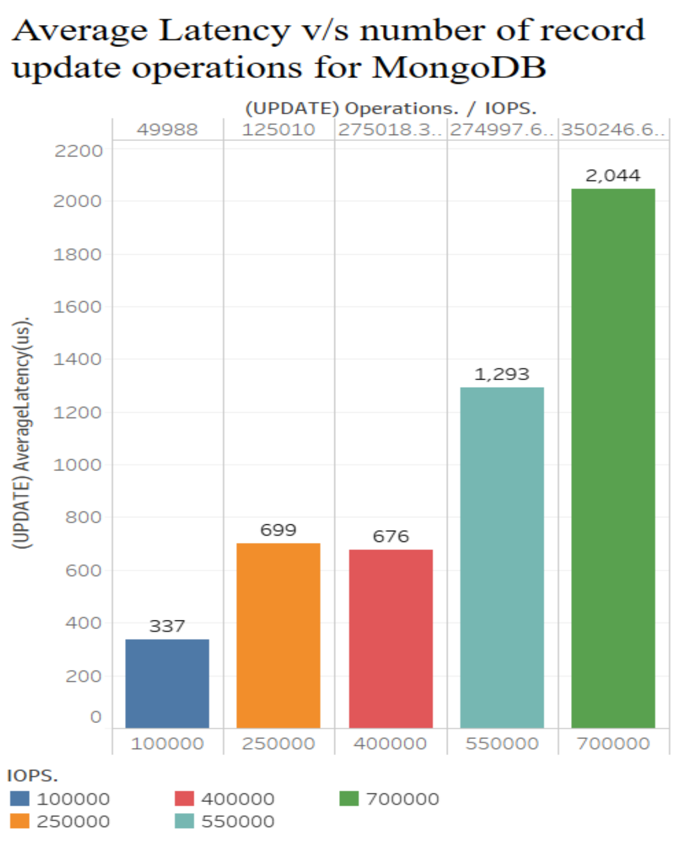




- For the opcount 100000, it can be illustrated that HBase leads in the average latency count that accounts for 264(us) with 50026 read operations as compared to 199(us) for MongoDB with 50012 read operations.
- For the opcount 250000, it can be stated that MongoDB has higher latency count that accounts for 566(us) with 124990 reads in contrast to the HBase that accounts for 455(us) with 125058.3 reads.
- For the opcount 400000, it can be explained that the MongoDB has more latency count that accounts for 1391(us) with 274981.6 reads in comparison to HBase that accounts for 961(us) with 199802 reads.
- For the opcount 550000, it can be described that the Hbase has highest latency count that accounts for 1391(us) with 274981.6 reads as compared to MongoDB that accounts for 1319(us) with 275002.3 reads.
- For the opcount 700000, it can be stated illustrated that HBase has highest latency value that accounts for 2647(us) with 350362 reads when compared to MongoDB that accounts for 1987(us) with 349753.3 reads.

2. A graphical representation of the recorded Average Latency against the number of record update operations for HBase and MongoDB over workload a,

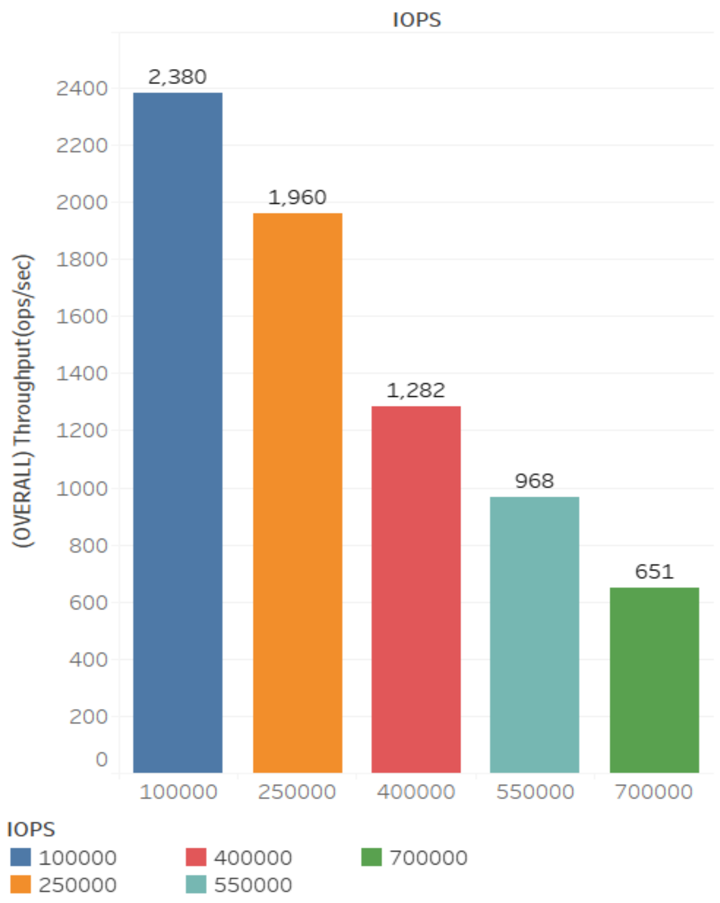




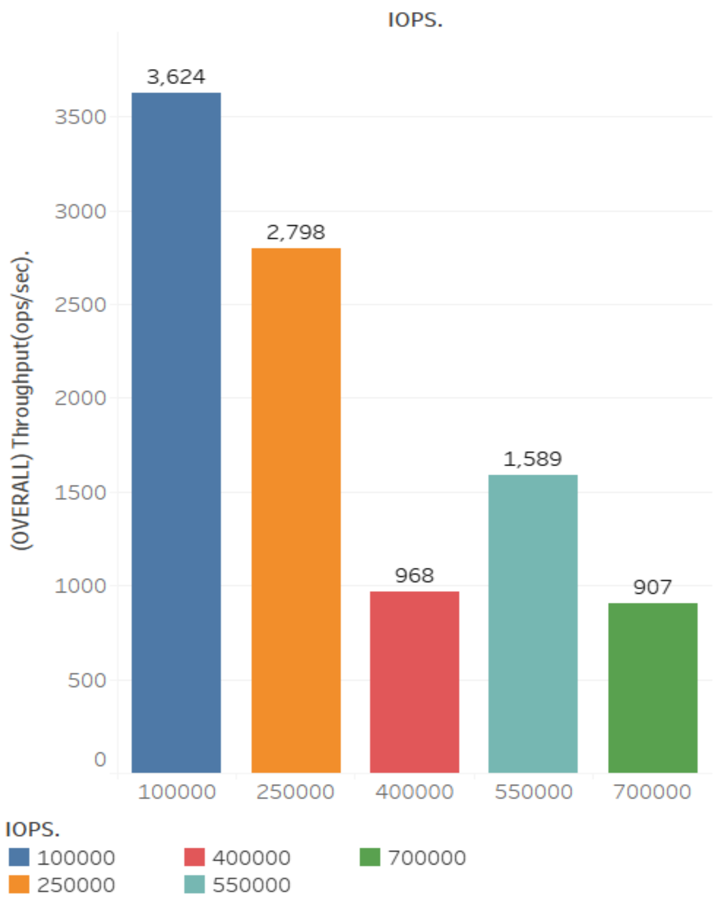
- For the opcount 100000, it can be illustrated that HBase leads in the average latency count that accounts for 538(us) with 49974 updates operations as compared to 337(us) for MongoDB with 49988 update operations.
- For the opcount 250000, it can be stated that MongoDB has higher latency count that accounts for 699(us) with 125010 updates in contrast to the MongoDB that accounts for 546.1 (us) with 124941.6 updates.
- For the opcount 400000, it can be explained that the MongoDB has more latency count that accounts for 676(us) with 275018.3 updates in comparison to HBase that accounts for 604.9(us) with 200198 updates.
- For the opcount 550000, it can be described that the MongoDB has the highest latency count that accounts for 1293(us) with 274997.6 updates as compared to HBase that accounts for 675.6(us) with 275018.3 updates.
- For the opcount 700000, it can be stated illustrated that MongoDB has the highest latency value that accounts for 2044(us) with 350246.6 updates when compared to HBase that accounts for 756.9(us) with 349637 updates.

3. A graphical representation of the recorded Overall Throughput against the total number of record operations for HBase and MongoDB over workload a,

Overall throughput v/s Total Record operations for HBase

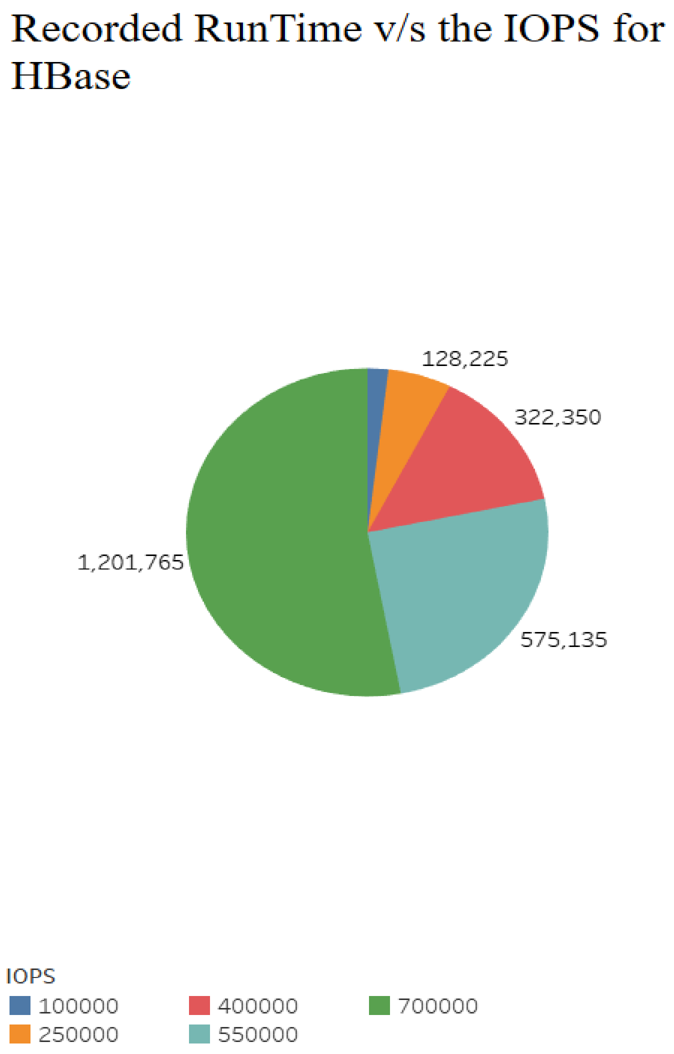


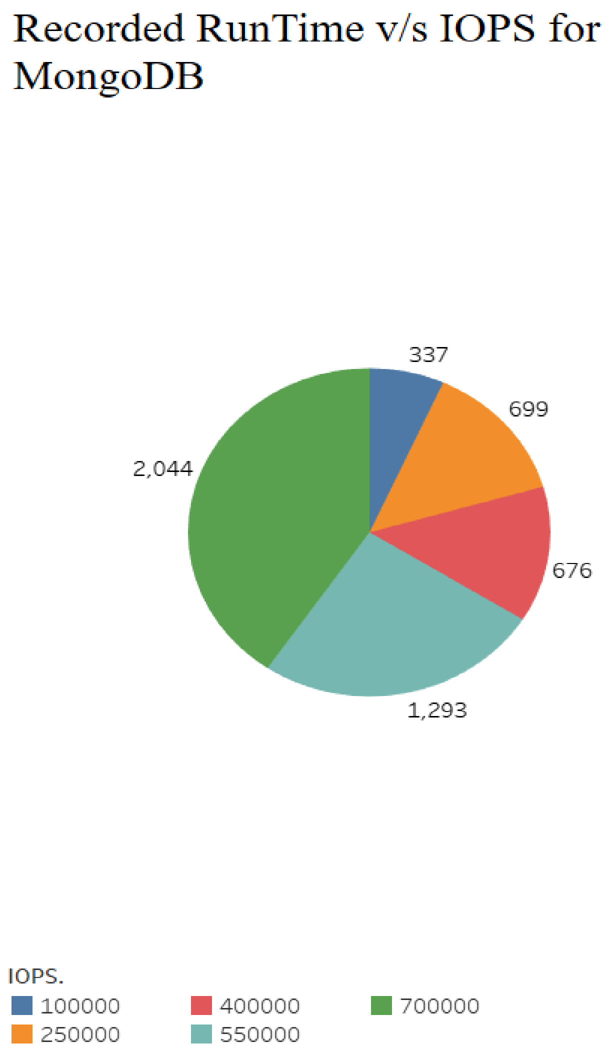
Overall throughput v/s Total Record operations for MongoDB



- For the opcount 100000, it can be illustrated that MongoDB leads in the throughput count that accounts for 3624(us) as compared to 2380(us) for HBase.
- For the opcount 250000, it can be stated that MongoDB has higher throughput count that accounts for 2798(us) in contrast to the HBase that accounts for 1960(us).
- For the opcount 400000, it can be explained that the HBase has more throughput count that accounts for 1282(us) in comparison to MongoDB that accounts for 968(us).
- For the opcount 550000, it can be described that the MongoDB has highest throughput count that accounts for 1589(us) as compared to HBase that accounts for 968(us).
- For the opcount 700000, it can be stated illustrated that MongoDB has highest throughput value that accounts for 907(us) when compared to HBase that accounts for 651(us).

4. A graphical representation of the recorded RunTime against the IOPS for HBase and MongoDB over workloads a,

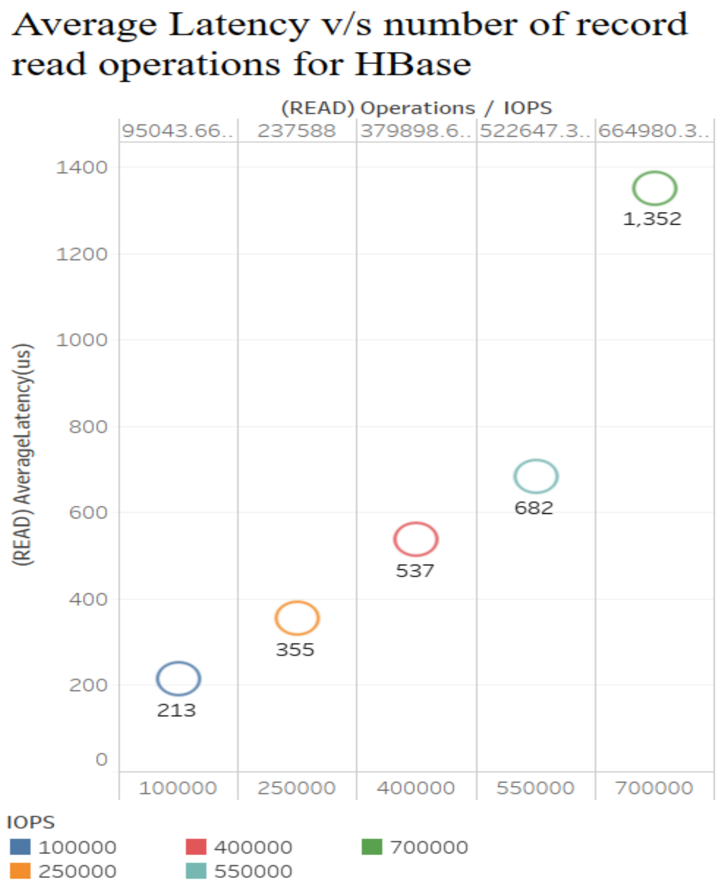


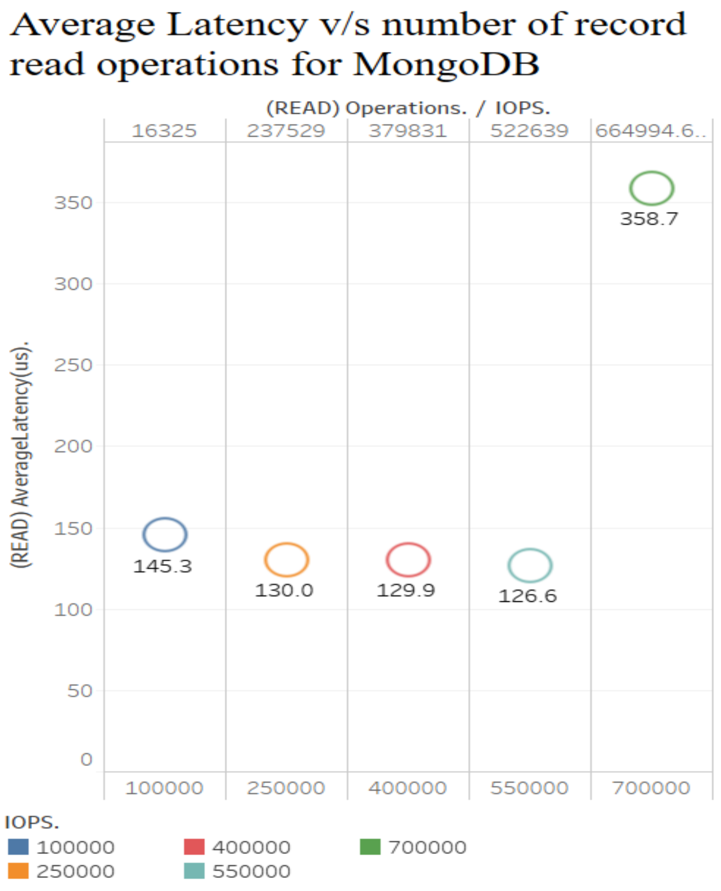


- For the opcount 100000, it can be illustrated that MongoDB has less run time that accounts for 337(ms) as compared to 42067(ms) for HBase.
- For the opcount 250000, it can be stated that MongoDB has less run time that accounts for 699(ms) in contrast to the HBase that accounts for 128225(ms).
- For the opcount 400000, it can be explained that the MongoDB has less run time that accounts for 676(ms) in comparison to HBase that accounts for 322350(ms).
- For the opcount 550000, it can be described that MongoDB has less run time that accounts for 1293(ms) as compared to HBase that accounts for 575135(ms).
- For the opcount 700000, it can be stated illustrated that MongoDB has less run time that accounts for 1293(ms) when compared to HBase that accounts for 1201765(ms).

b) WORKLOAD D :

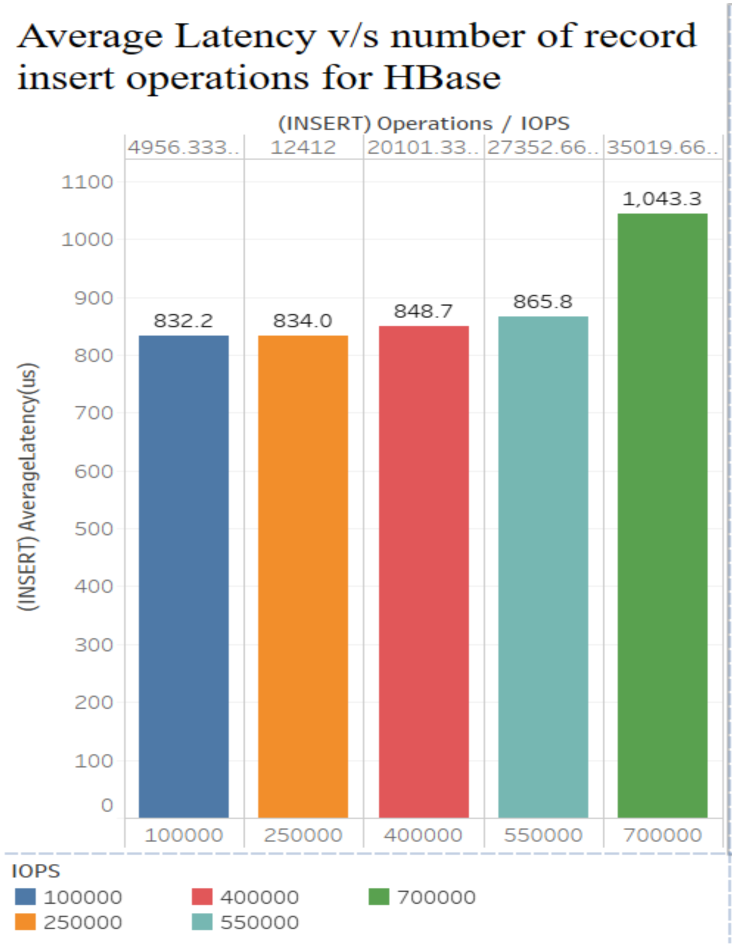
1. A graphical representation of the recorded Average Latency against the number of record read operations for HBase and MongoDB over workload d,

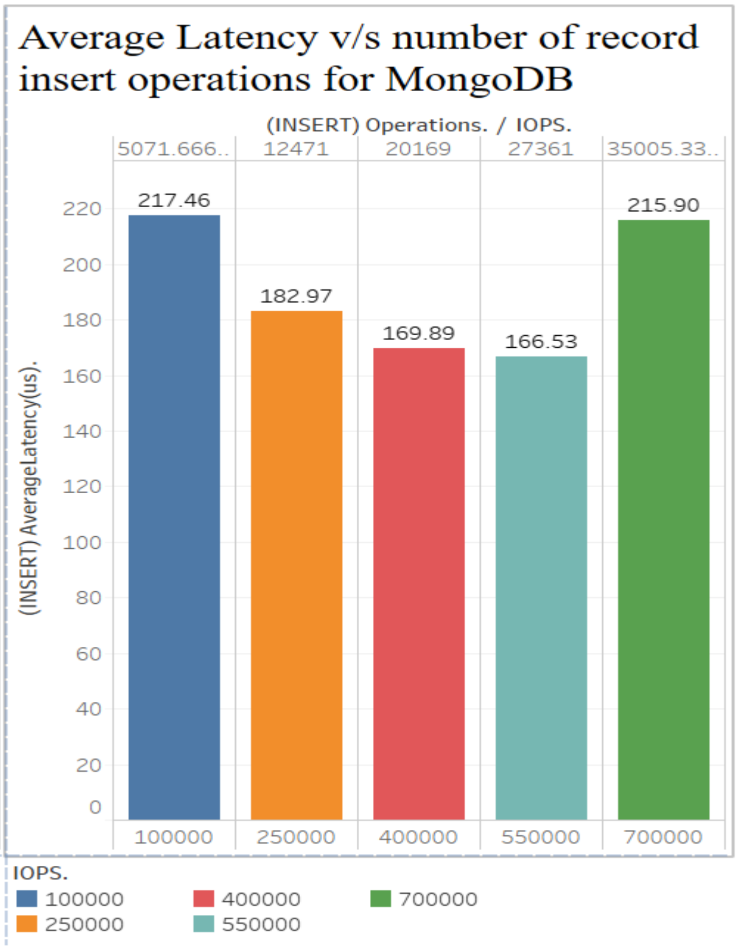




- For the opcount 100000, it can be illustrated that HBase leads in the average latency count that accounts for 213(us) with 95043.66 read operations as compared to 145.3(us) for MongoDB with 16325 read operations.
- For the opcount 250000, it can be stated that HBase has higher latency count that accounts for 355(us) with 237588 reads in contrast to the MongoDB that accounts for 130(us) with 237529 reads.
- For the opcount 400000, it can be explained that the HBase has more latency count that accounts for 537(us) with 379898.6 reads in comparison to MongoDB that accounts for 129.9(us) with 379831 reads.
- For the opcount 550000, it can be described that the HBase has highest latency count that accounts for 682(us) with 522647.3 reads as compared to MongoDB that accounts for 126.6(us) with 522639 reads.
- For the opcount 700000, it can be stated illustrated that HBase has highest latency value that accounts for 1352(us) with 664980.3 reads when compared to MongoDB that accounts for 358.7(us) with 664994.6 reads.

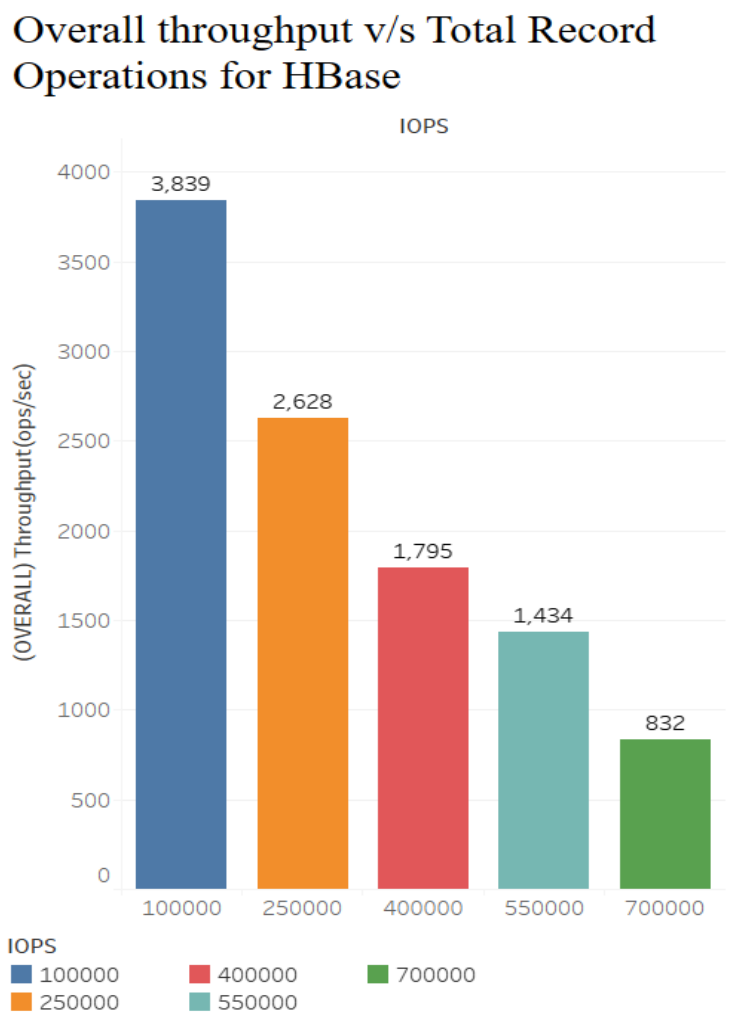
2. A graphical representation of the recorded Average Latency against the number of record insert operations for HBase and MongoDB over workload d,

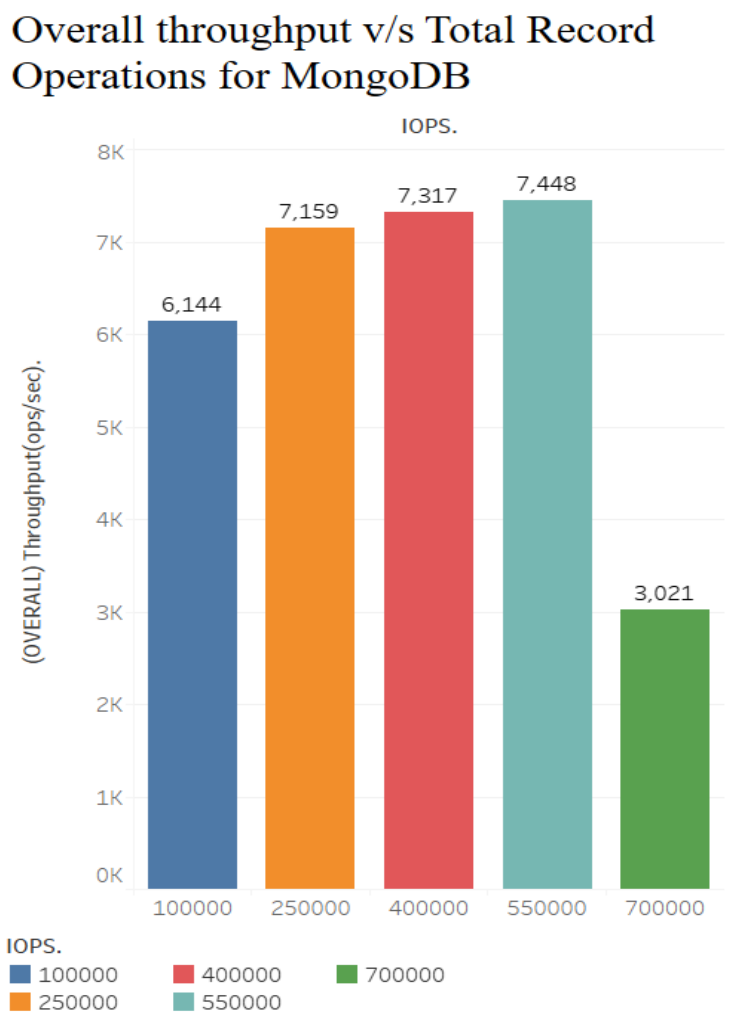




- For the opcount 100000, it can be illustrated that HBase leads in the average latency count that accounts for 832.2(us) with 4956.33 insert operations as compared to 217.46(us) for MongoDB with 5071.66 insert operations.
- For the opcount 250000, it can be stated that HBase has higher latency count that accounts for 834(us) with 12412 inserts in contrast to the MongoDB that accounts for 182.97 (us) with 12471 inserts.
- For the opcount 400000, it can be explained that the HBase has more latency count that accounts for 848.7(us) with 20101.33 inserts in comparison to HBase that accounts for 169.89(us) with 20169 inserts.
- For the opcount 550000, it can be described that the HBase has the highest latency count that accounts for 865.8(us) with 27352.66 inserts as compared to HBase that accounts for 166.53(us) with 27361 inserts.
- For the opcount 700000, it can be stated illustrated that HBase has the highest latency value that accounts for 1043.3(us) with 35019.66 inserts when compared to MongoDB that accounts for 215.90(us) with 35005.33 inserts.

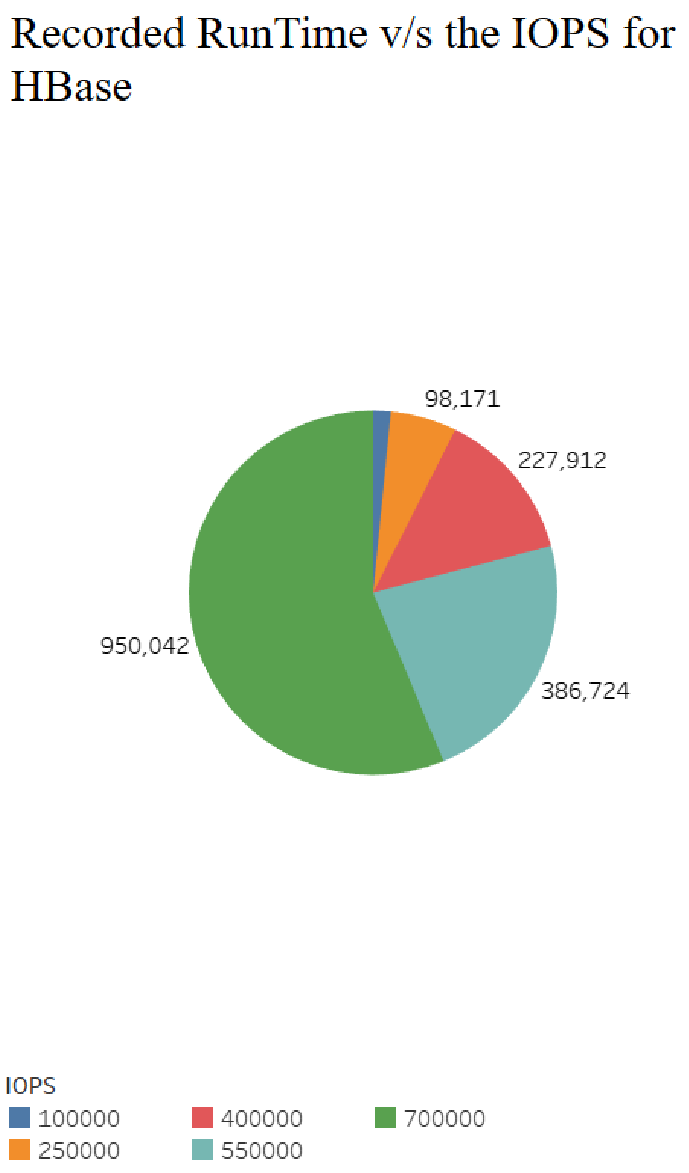
3. A graphical representation of the recorded Overall Throughput against the total number of record operations for HBase and MongoDB over workload d,

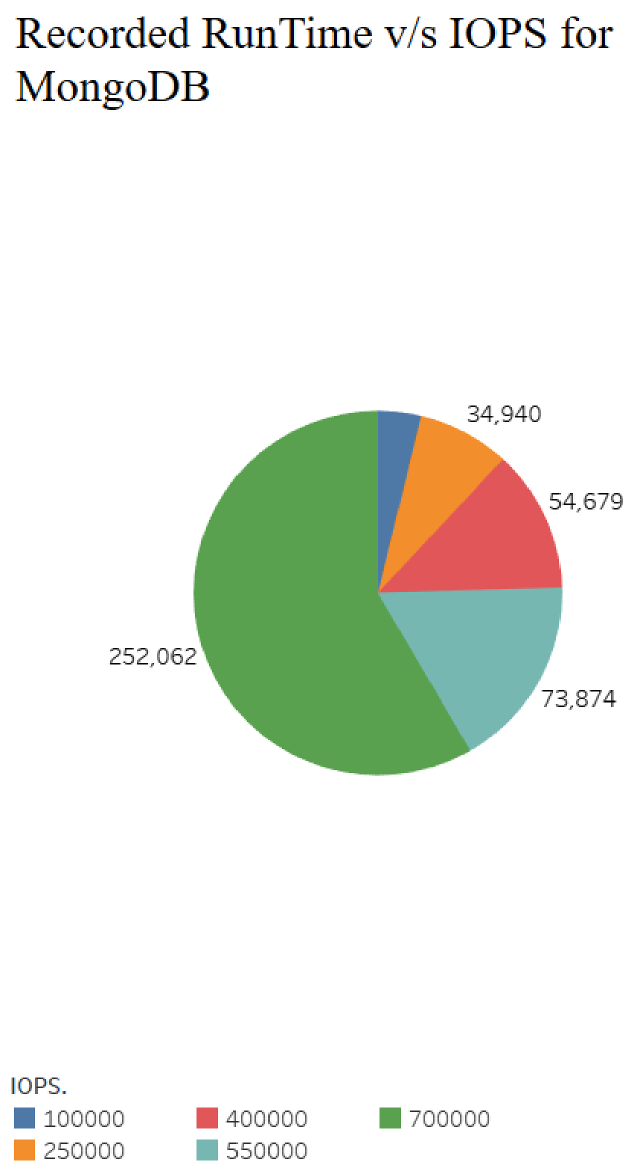




- For the opcount 100000, it can be illustrated that MongoDB leads in the throughput count that accounts for 6144(us) as compared to 3839(us) for HBase.
- For the opcount 250000, it can be stated that MongoDB has higher throughput count that accounts for 7159(us) in contrast to the HBase that accounts for 2628(us).
- For the opcount 400000, it can be explained that the MongoDB has more throughput count that accounts for 7317(us) in comparison to HBase that accounts for 1795(us).
- For the opcount 550000, it can be described that the MongoDB has highest throughput count that accounts for 7448(us) as compared to HBase that accounts for 1434(us).
- For the opcount 700000, it can be stated illustrated that MongoDB has highest throughput value that accounts for 3021(us) when compared to HBase that accounts for 832(us).

4. A graphical representation of the recorded RunTime against the IOPS for HBase and MongoDB over workloads d,





- For the opcount 100000, it can be illustrated that MongoDB has less run time that accounts for 16325(ms) as compared to 26056(ms) for HBase.
- For the opcount 250000, it can be stated that MongoDB has less run time that accounts for 34940(ms) in contrast to the HBase that accounts for 98171(ms).
- For the opcount 400000, it can be explained that the MongoDB has less run time that accounts for 54679(ms) in comparison to HBase that accounts for 227912(ms).
- For the opcount 550000, it can be described that MongoDB has less run time that accounts for 73874(ms) as compared to HBase that accounts for 386724(ms).
- For the opcount 700000, it can be stated illustrated that MongoDB has less run time that accounts for 252062(ms) when compared to HBase that accounts for 950042(ms).

VIII. Conclusion and discussion

The issues faced by most of the organizations to manage a heavy burden of daily data has been addressed in this research paper. Earlier, the organizations used to work with the relational databases that had many limitations associated with it. Thus, it had a major drawback when it comes to scaling the system to accommodate more data. The overloading of data resulted in system crashes and failures. The organizations had to step back when the upgradations in their infrastructure demanded heavy tolerance of data. Thus, to counter this downside, NoSQL databases were brought into picture. This shown a great increased trend lines in terms of adapting to these NoSQL based database systems. Thus, this led to more and more rise in the number of such databases. Hence, there arose a need to categorize them as per their area of expertise for storing different forms of data and more importantly based on their individual performances. To accomplish an effective comparison between these databases, there have been several performance benchmarking tools available. For this research, YCSB has been utilized as it is one of the best suited tool for measuring the performance on various measures. Thus, the primary aim of this research paper have been successfully achieved. Both the databases have been tested by applying the same workloads and opcounts using YCSB and testharness. The fluctuations in the latency values of the reads, updates and inserts of the records directly affects the throughput in the graphs. Therefore, from the above evaluated output, it can be concluded that MongoDB outperformed HBase in almost all the tests. There is an observable difference between both the database results. Thus, MongoDB can be stated as an efficient system than HBase in terms of the performance measures discussed in the paper.

IX. References

Brain4ce Education Solutions, 2014. *HBase Architecture: HBase Data Model & HBase Read/Write Mechanism.* [Online] Available at: https://www.edureka.co/blog/hbase-architecture/ [Accessed 2018].

Anon., 2008. *Apache HBase.* [Online] Available at: https://en.wikipedia.org/wiki/Apache_HBase [Accessed December 2018].

datadiversity, year. *A Brief History of Non-Relational Databases.* [Online] Available at: http://www.dataversity.net/a-brief-history-of-non-relational-databases/ [Accessed 2018].

Dezyre, n.d. *Overview of HBase Architecture and its Components.* [Online] Available at: https://www.dezyre.com/article/overview-of-hbase-architecture-and-its-components/295 [Accessed 2018].

EnqingTang, Y. F., 2016. Performance Comparison between Five NoSQL Databases. *IEEE,* pp. 1-5.

Foote, K. D., 2018. *A Brief History of Non-Relational Databases.* [Online] Available at: http://www.dataversity.net/a-brief-history-of-non-relational-databases/ [Accessed 2018].

George Seriatos, G. K. A. M. D. K. T. V., 2016. Comparison of Database and Workload types and performance in Cloud Environments. *Springer International Publishing,* pp. 1-13.

Guru99, 2018. *HBase Tutorials for Beginners.* [Online] Available at: https://www.guru99.com/hbase-tutorials.html [Accessed 2018].

Guru99, 2018. *MongoDB Tutorial for Beginners.* [Online] Available at: https://www.guru99.com/mongodb-tutorials.html [Accessed 2018].

Jagdev Bhogal, I. C., 2015. Handling Big Data using NoSQL. *IEEE,* pp. 1-6.

jaiswal, s., 2018. *HBase Tutorial.* [Online] Available at: https://www.javatpoint.com/hbase [Accessed 2018].

MATEI, L., 2014. Big Data Issues: Performance, Scalability, Availability. *Journal of Mobile, Embedded and Distributed Systems,* Volume 1, p. 1.

Pijin Gong, C. L. Y. G. H. M. Y. S. L. W., 2017. Research on Keyword Retrieval Method of HBase Database. *American Institute of Physics.*

Raj, S., 2015. Storing of Unstructured data into MongoDB using Consistent Hashing. *researchgate,* pp. 1-47.

researchgate, 2015. Storing of Unstructured data into MongoDB. *researchgate.*

researchgate, 2018. *Correlating NoSQL Databases With a Relational Database: Performance and Space.* [Online] [Accessed 2018].

Rohini Gaikwad, A. C. G., 2015. A Study of YCSB –tool for measuring a performance of NOSQL databases. *IJETCR,* Volume 3, pp. 1-4.

Suman Kashyap, S. Z. T. B. S. S., 2013. Benchmarking and Analysis of NoSQL Technologies. *IJETAE,* September, 3(9), pp. 1-5.

wideskills, 2015. *MongoDB Architecture.* [Online] Available at: http://www.wideskills.com/mongodb-tutorial/03-mongodb-architecture [Accessed 2018].

wikipedia, 2009. *MongoDB.* [Online] Available at: https://en.wikipedia.org/wiki/MongoDB [Accessed 2018].

Yusuf Abubakar, T. S. A. I. G. A., 2014. Performance Evaluation of NoSQL Systems Using YCSB. *IJAIS,* September , Volume 7, pp. 1-5.